VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Adaptace parametrů samoorganizujícího se migračního algoritmu pomocí neuronových sítí

On the Self-Organizing Migrating Algorithm Parameters Adaptation Using Neural Networks

2020 Tatiana Oškrobaná

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Tatiana Oškrobaná**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Adaptace parametrů samoorganizujícího se migračního algoritmu pomocí neuronových sítí
On the Self-Organizing Migrating Algorithm Parameters Adaptation Using Neural Networks

Jazyk vypracování: slovenština

Zásady pro vypracování:

Samoorganizující se migrační algoritmus (SOMA) [1] je metaheuristický algoritmus inspirovaný chováním zvířat. Původní algoritmus byl již několikrát vylepšen a vznikly tak nové, lepší a rychlejší varianty tohoto algoritmu [2], [3]. Dalšího vylepšení by mohlo být dosaženo využitím adaptivních parametrů. Úkolem této práce je ověřit, zda k tomuto účelu může být využita neuronová síť [4]. Vstupem této sítě mohou být informace o populaci [5], fáze, ve které se algoritmus nachází, nebo vzdálenost mezi jedinci. Výstupem pak budou jednotlivé parametry SOMA algoritmu.

Úkoly této práce se dají shrnout v těchto bodech:
1. Nastudovat problematiku SOMA algoritmu.
2. Nastudovat problematiku neuronových sítí.
3. Naprogramovat SOMA algoritmus.
4. Navrhnou a natrénovat neuronovou síť pro adaptaci parametrů.
5. Porovnat klasický SOMA algoritmus s novým algoritmem využívajícím neuronovou síť.

Seznam doporučené odborné literatury:

[1] DAVENDRA, Donald, et al. Self-organizing migrating algorithm. New optimization techniques in engineering, 2016.
[2] DOS SANTOS COELHO, Leandro; MARIANI, Viviana Cocco. An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect. Energy Conversion and Management, 2010, 51.12: 2580-2587.
[3] DIEP, Quoc Bao. Self-Organizing Migrating Algorithm Team To Team Adaptive–SOMA T3A. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2019. p. 1182-1187.
[4] OJHA, Varun Kumar; ABRAHAM, Ajith; SNÁŠEL, Václav. Metaheuristic design of feedforward neural networks: A review of two decades of research. Engineering Applications of Artificial Intelligence, 2017, 60: 97-116.
[5] OLORUNDA, Olusegun; ENGELBRECHT, Andries P. Measuring exploration/exploitation in particle swarms using swarm diversity. In: 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence). IEEE, 2008. p. 1128-1134.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Lukáš Tomaszek**

Datum zadání: 01.09.2019

Datum odevzdání: 30.04.2020

doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne. Uviedla som všetky literárne pramene a publikácie, z ktorých som čerpala.

V Zborove nad Bystricou 6. máj 2020 .....

Rada by som na tomto mieste poďakovala všetkým, ktorí mi s prácou pomohli, pretože bez nich by táto práca nevznikla. V prvom rade by som sa chcela poďakovať vedúcemu tejto práce Ing. Lukáši Tomaszkovi za odborné rady a pomoc vždy, keď som ju potrebovala. Ďalej by som sa chcela poďakovať svojim rodičom a bratovi so sestrou za podporu a pomoc s vecami, na ktoré som nemala čas.

Abstrakt

Za posledné roky došlo k mnohým vylepšeniam týkajúcich sa Samo-Organizujúceho sa Migračného Algoritmu. Cieľom mojej bakalárskej práce bolo zistiť, či je možné vylepšiť tento algoritmus pridaním neurónovej siete, aby sme dostali nový algoritmus, ktorý bude efektívnejšie pracovať tak, že bude pre každého jedinca jednotlivo vyberať nastavenia parametrov počas migračného kola. Neurónová sieť sa učí pomocou Samo-Organizujúceho sa Migračného algoritmu. Nový algoritmus má rozhodovať o hodnotách jednotlivých parametrov pre vybraného jedinca v migračných kolách. Ukázalo sa, že neurónová sieť tieto hodnoty nastavuje veľmi podobne u jednotlivých funkcií. Z výsledných hodnôt sme zistili, že algoritmus spolu s neurónovou sieťou dosahuje podobné až lepšie výsledky na jednotlivých testovacích funkciách.

Kľúčové slová: SOMA, neurónová sieť, adaptívny, samo-organizujúci sa, metaheuristické algoritmy

Abstract

In recent years, there have been many improvements regarding the Self-Organizing Migrating Algorithm. The aim of my bachelor thesis was to find out if it is possible to improve this algorithm by adding a neural network to get a new algorithm that will work more efficiently by selecting individual parameter settings for each individual during the migration loop. The neural network is learned by using the Self-Organizing Migrating Algorithm. The new algorithm should decide on the values of individual parameters for the selected individual. It turned out that the neural network sets these values very similarly for the specific functions. From the resulting values, we found that the algorithm together with the neural network achieves similar to better results on individual test functions.

Keywords: SOMA, neural network, adaptive, self-organizing, metaheuristic algorithms

Obsah

Zoznam použitých skratiek a symbolov

SOMA	–	Samo-Organizujúci sa migračný algoritmus
SOMA_L	–	SOMA použitý na učenie neurónovej siete
SOMANN	–	SOMA v spojení s neurónovou sieťou
GA	–	Genetický algoritmus
DSOMA	–	Discrete SOMA
MOSOMA	–	Multi-Objective SOMA
ES	–	Evolučné Stratégie

Zoznam obrázkov

Zoznam tabuliek

1 Úvod

Biologicky inšpirované algoritmy sa stávajú za posledné obdobie viac a viac diskutovanou témou. Sú schopné hľadať riešenia na problémy, ktoré nedokážeme vyriešiť ľudskými schopnosťami. Dokážu niekedy uspieť aj tam, kde iné algoritmy zlyhajú, sú určené na zložité optimalizačné úlohy. Je na nich zaujímavé to, že riešenia na mnohé komplikované problémy dokážeme nájsť pomocou inšpirovaním sa prirodzeným chovaním zvierat. Ukrýva sa v ich systéme istá sofistikovanosť a prinášajú nový pohľad, hoci sú tieto spôsoby dávno zaužívané a môžeme ich pozorovať od nepamäti.

Využívajú zároveň dva pohľady na správanie sa zvierat a ich biologické princípy. Môžeme sa pozerať na zvieracie spoločenstvo z pohľadu, že spolu dokážu jedinci spolupracovať a spoločne dosiahnuť cieľ, z ktorého budú individuálne profitovať všetci jedinci spoločenstva. Takéto správanie sa môžeme sledovať u včiel či rýb. Iný pohľad nám prinášajú zvieratá, ktoré spolu súperia a vyhrať chce ten najlepší, kde ide prevažne o súťaživé správanie. Oba tieto spomínané prístupy využíva Samo-Organizujúci sa Migračný Algoritmus (SOMA), ktorý spája spoluprácu so súperením. Kombináciou obidvoch pohľadov sa stal účinným pri riešení mnohých optimalizačných problémov. Odkedy bol algoritmus v roku 2004 uvedený Zelinkom [1], prešiel mnohými vylepšeniami a snahou ho zefektívňovať. Stalo sa tak aj vo veľmi krátkej minulosti a algoritmus neprestáva byť zaujímavou témou ani za čas, odkedy bol uvedený. Stále je dostatočne veľký priestor na zdokonaľovanie či vylepšovanie algoritmu.

Mojou úlohou bolo zistiť, či tento algoritmus môže pracovať efektívnejšie pridaním neurónovej siete [2]. Neurónové siete sú tiež veľmi silným nástrojom na pomoc pri riešení úloh a snažia sa čo najlepšie napodobniť biologické neurónové siete, ktorými sú tie umelé inšpirované. Neurónové siete sa využívajú v mnohých odvetviach. Dokážu byť úspešné pri klasifikácii [3], predikcii [4] či simulovaní pamäte [5]. Sú schopné aproximovať akúkoľvek spojitú funkciu a nevyžadujú informácie o štruktúre procesu, na ktoré ich aplikujeme.

Neurónová sieť má algoritmus SOMA pomôcť vylepšiť, aby v spojení našli nové, prípadne lepšie riešenia. Vzniknuté spojenie SOMA a neurónovej siete (SOMANN) sa učí za pomoci SOMA$_L$. Úlohou neurónovej siete v algoritme SOMANN je určiť nastavovanie parametrov pre každého jedinca individuálne, podľa toho, ako daný bude potrebovať toto nastavenie zmeniť. Učenie neurónovej siete pomocou SOMA$_L$ algoritmu je prevedené tak, že SOMA$_L$ sieť optimalizuje a hľadá pre ňu čo najlepšie váhy, ktoré sieť použije pri rozhodovaní o parametroch jedinca. Algoritmus SOMANN priniesol opäť nový pohľad na to, ako SOMA môže byť vylepšovaný. Do neurónovej siete vstupujú informácie o jedincovi a populácii a neurónová sieť podľa toho určí, aké nastavenie parametrov bude vhodné pre vybraného jedinca a jedinec už vykonáva migračné kola s parametrami, ktoré mu navrhla sieť.

V jednotlivých kapitolách popisujem SOMA algoritmus a základnú charakteristiku biologicky inšpirovaných algoritmov a ich rozdelenie. Ďalej predstavujem modifikácie alebo vylepšenia SOMA algoritmu, odkedy bol uvedený a popisujem, k čomu prispeli jednotlivé objavy ohľadom

tohto algoritmu. V ďalšej kapitole predstavujem umelé neurónové siete a ich charakteristiky. Venujem sa základným typom neurónových sietí, ich popisu a uvedenie hlavných typov aktivačných funkcií. Ďalej sa venujem popisu nového algoritmu SOMANN, ktorý vznikne spojením neurónovej siete a pôvodného SOMA algoritmu. Na konci potom oba testujem na vybraných testovacích funkciách a porovnávam ich štatistické údaje.

2 Biologicky inšpirované algoritmy

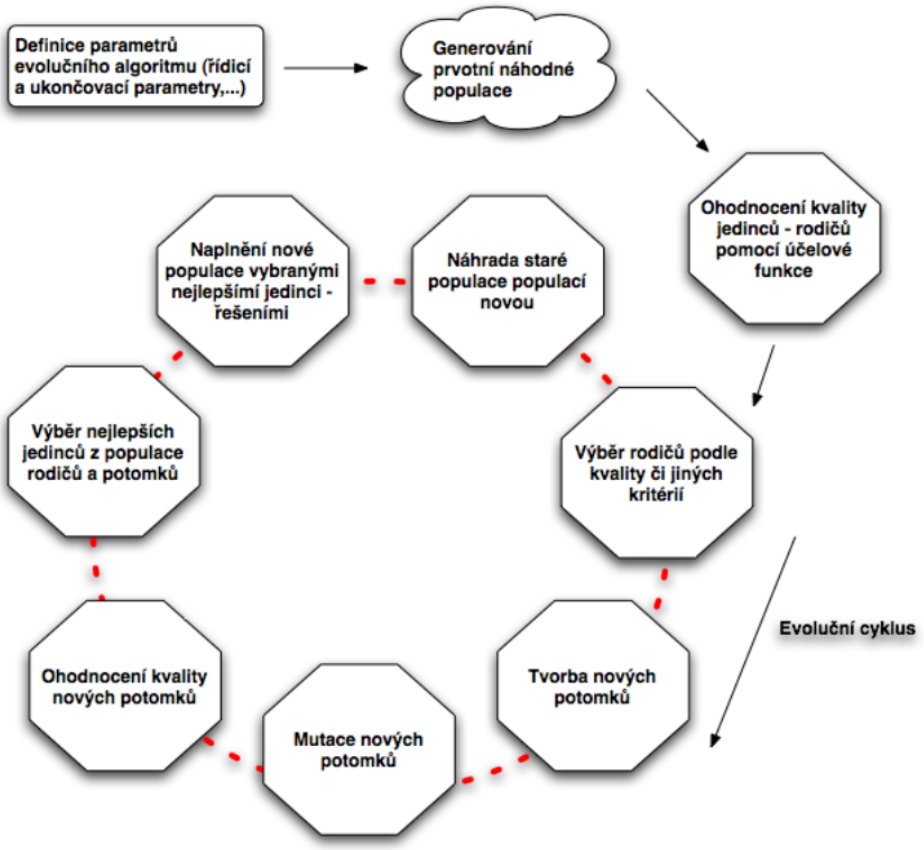
Biologicky inšpirované algoritmy sú založené na princípoch inšpirovaných biologickou evolúciou v prírode na rozvoj nových, robustných techník optimalizácie. Problémy, ktorými sa zaoberajú sú väčšinou nelineárne a preto využívajú veľkú dimenziu a sú časovo náročné. Výskum za posledné roky ukázal, že reprezentujú sľubné riešenie na komplexné optimalizačné problémy. Využívajú sa hlavne v odvetviach ako matematika, biológia či informačné technológie [6]. Do tejto skupiny môžeme zaradiť SOMA algoritmus s jeho súťaživo-kooperatívnym chovaním jedincov, ale i umelé neurónové siete, ktoré sú inšpirované ľudskou inteligenciou.

2.1 Evolučné algoritmy

Evolučné algoritmy pracujú s populáciou potencionálnych riešení, ktoré sa vyvíjajú v generáciách, z ktorých sú vyberané najlepšie alebo individuálne najvhodnejšie riešenia. Podľa Darwinovej a Mendelovej teórie evolúcie sú z rodičov plodení potomkovia tak, že pri svojom vzniku podliehajú mutáciám. Po generácie rodičia a potomkovia nevhodný pre aktuálne životné prostredie vymierajú, aby uvoľnili miesto lepším potomkom. Princíp evolúcie je znázornený na Obr.1.

Autor SOMA algoritmu, Zelinka, zaraďuje tento algoritmus medzi algoritmy evolučné. SOMA je klasifikovaný ako evolučný algoritmus, pretože finálny výsledok, po migračnom kole, je ekvivalentom výsledku jednej generácie, ktorú môžeme chápať ako novú pozíciu v N-dimenzionálnom hyper priestore. Pri vytvorení populácie, pravidlá riadenia všetkých jedincov predstavuje „samoorganizujúce sa“ správanie [1].

Medzi nové stochasticky založené techniky na riešenie inžinierskych problémov pomocou informačných technológií patria Genetické algoritmy (GA), Evolučné Stratégie (ES) [7] či Difereniálna Evolúcia (DE) [8]. Sú založené na evolučnej teórii, čiže obsahujú znaky ako mutácie, kríženie, selekcia, prežitie či vhodnosť. Evolučné algoritmy kombinujú všetky adaptívne a výpočtové modely prírodného evolučného systému - GA, ES, evolučné a genetické programovanie. Sú vhodné pri hľadaní globálneho riešenia optimalizačných problémov a sú jednoduché na implementáciu.



Obr. 1: Evolučný princíp [1]

2.2 Memetické algoritmy

Správanie sa zvieracích skupín v prírode, ako napr. hmyzu, kŕdľov vtákov či kolónií včiel alebo svorky vlkov, je inšpiráciou vo výpočtovej technike na riešenie mnohých vedeckých či inžinierskych problémov. Koncept memetických algoritmov je založený na simulovaní zjednodušeného sociálneho systému. Sú založené na kolektívnom chovaní rôznych spoločenstiev. Ich vývoj pozostáva z pozorovania správania sa zvieracích spoločenstiev v určitých situáciách. Väčšina súčasných algoritmov je realizovaná na optimalizačné úlohy [9]. Simulovanie pohybu či správania sa jedincov v prírode prináša spôsob ako takéto problémy riešiť. Znakmi memetických algoritmov sú napr. adaptácia, rozšíriteľnosť, rýchlosť či paralerizmus. Každý jedinec zo zvieracej skupiny je individuálne zodpovedný za špecifickú úlohu, niekedy jedinci spolupracujú na dosiahnutie spoločného cieľa. Memetické algoritmy sú schopné pracovať v n-dimenzionálnom priestore a vyznačujú sa tým, že sú schopné nájsť výsledok blížiaci sa požadovanému za pomerne krátku dobu. SOMA je inšpirovaný memetickými algoritmami a využíva pri riešení problémov pohyb, ktorý využívajú aj memetické algoritmy.

2.3 Heuristické algoritmy

Slovo „heuristics“ znamená hľadať, objavovať za pomoci experimentov, aj zlyhaní. Heuristiky dokážu nájsť kvalitné riešenie na optimalizačné problémy za rozumný čas, ale nedokážu garantovať optimálne riešenie. Hodia sa pokiaľ nutne nie je potrebné najlepšie riešenie, ale stačí prijateľne dobré [10].

Heuristické algoritmy sa delia na deterministické a stochastické. V tomto delení zaraďujeme SOMA algoritmus do skupiny stochastických algoritmov, čo znamená že niektoré kroky využívajú náhodné operácie, čiže výsledky rovnakých riešení, ktoré nimi dostaneme, sa môžu líšiť v jednotlivých behoch programu. V takýchto prípadoch má potom zmysel spustiť program viackrát a vybrať najlepšie získané riešenie.

2.4 Metaheuristiky

Vo výpočtovej technike boli metaheuristiky navrhnuté na hľadanie riešení daných komplexnými problémami, ktoré bolo potrebné optimalizovať. Metaheuristické optimalizačné algoritmy sú simulované alebo inšpirované správaním skupiny zvierat. Metaheuristické optimalizačné algoritmy sú známe kvôli svojej dobrej schopnosti nájsť optimálne riešenie na daný problém, sú ľahko implementovateľné a sú aplikovateľné na rôzne odvetvia [6]. Riešia viacparametrové optimalizačné problémy s mnohými lokálnymi extrémami a neznámym gradientom. Všetky metaheuristiky využívajú nejakú formu randomizácie alebo lokálneho vyhľadávania. Randomizácia prináša dobrý spôsob ako sa vyhnúť iba lokálnemu hľadaniu a zachovať hľadanie na globálnej úrovni. Keďže sa tieto metódy často inšpirujú prírodnými procesmi a javmi, môžeme ich zaraďovať do skupiny algoritmov evolučných [10].

Zaraďujeme tu algoritmus Artificial Bee Colony (ABC) [11], ktorý simuluje spoluprácu kolónie včiel, ďalej sem radíme algoritmus Particle Swarm Optimization (PSO) [12], ktorý simuluje správanie zvieracích spoločenstiev, ako napr. rýb či vtákov, ďalej grey wolf optimizer (GWO) [13], ktorý simuluje techniky lovu a vedenie svorky vlkov, či Lion Optimization Algorithm (LOA) [14], ktorý napodobňuje správanie levov a ich charakteristické správanie ohľadom spolupráce.

3 SOMA

Samo-Organizujúci sa Migračný Algoritmus (SOMA) bol prezentovaný Zelinkom v roku 2004 [1]. Algoritmus je založený na vzájomnej spolupráci jedincov pri riešení spoločného problému. Jedinci medzi sebou súťažia, ale zároveň spolu kooperujú za účelom dosiahnutia rovnakého cieľa. Jedná sa o chovanie, ktoré môžeme pozorovať hocikde v prírode - chovanie sa nejakého stáda alebo svorky za účelom získania potravy, jedinci si medzi sebou vymieňajú informácie a tak rýchlejšie a efektívnejšie dôjdu k riešeniu spoločného problému.

Proces hľadania potravy dokážeme rozdeliť na dve fázy, prvá fáza je fáza súťaživá - každý jedinec sa samostatne snaží nájsť najlepší zdroj potravy a vyhrať nad ostatnými, ktorí majú rovnakú úlohu. Po ukončení prvej fázy prejdú jedinci do druhej fázy, čiže fázy spolupráce - tu si medzi sebou vymenia informácie získané hľadaním zdroja. Zvyšok jedincov bude nasledovať toho, ktorý našiel lepší zdroj ako ostatní. Takéto chovanie sa môže opakovať až dokým nepôjdu za jedincom s najlepším zdrojom. Jedná sa o princíp súťaživo-kooperatívneho chovania. Algoritmus je mnohokrát schopný nájsť aj globálny extrém, v uvedenom príklade je to zdroj potravy [15].

Keďže je SOMA založená na súťaživo-kooperatívnom princípe, jeho varianty sa teda nazývajú stratégie. Jedinci sa rôzne pohybujú a vzájomne ovplyvňujú. Základná stratégia, „Všetci k jednému“, z anglického „AllToOne“, sa skladá z krokov popísaných nižšie.

3.1 Definícia parametrov

Na začiatku definujeme riadiace a ukončovacie parametre, tie sú užívateľsky zvolené. Pred začatím sa definuje účelová funkcia (*cost function*) alebo tzv. vhodnosť (*fitness*) - upravená návratová hodnota účelovej funkcie. Účelová funkcia je matematický model problému, ktorého maximalizáciou alebo minimalizáciou prídeme k hľadanému riešeniu problému.

Algoritmus SOMA má takéto parametre [16]:

- Veľkosť populácie (*PopSize*) - Počet jedincov populácie.
- Dimenzia (D) - Počet parametrov jedincov.
- Dĺžka cesty (*PathLenght*) - Určuje ako ďaleko sa zastaví aktívny jedinec od lídra. Ak je hodnota $PathLenght = 1$, tak aktívny jedinec zastaví jeho pohyb na lídrovej pozícii.
- Skok (*Step*) - Jeden krok ukazuje ako sa postupne aktívny jedinec posúva po zvolenej trase.
- Pertubácia (PRT) - Číslo, podľa ktorého sa generuje *PRTVector*.
- Počet migračných kôl (*MigrationsCount*) - Číslo, ktoré znamená koľkokrát sa populácia preorganizuje.

Rozsah a určenie jednotlivých parametrov je v Tabuľke 1.

Parameter	Doporučený rozsah	Poznámka
PathLength	[1,1; 5>]	Riadiaci parameter
Step	[0,11; PathLength]	Riadiaci parameter
PRT	[0; 1]	Riadiaci parameter
D	dané problémom	Počet argumentov účelovej funkcie
PopSize	[10; definuje užívateľ]	Riadiaci parameter
MigrationsCount	[10; definuje užívateľ]	Ukončovací parameter

Tabuľka 1: SOMA parametre [1]

3.2 Tvorba populácie

Populácia je tvorená pomocou generátora náhodných čísel. Do každého parametra jedinca je priradené náhodné číslo v rozsahu $[x_j^{Max}, x_j^{Min}]$, ktoré určujú jeho typ a hodnotu.

Populácia je vygenerovaná pomocou vzťahu:

$$P^{(0)} = x_{i,j}^{(0)} = rnd_j[0.1] \cdot (x_j^{Max} - x_j^{Min}) + x_j^{Min},$$

kde $P^{(0)}$ predstavuje prvopočiatočnú populáciu,
$x_{i,j}$ je j-tý parameter i-tého jedinca,
x_j^{Max} je horná hranica j-tého parametra,
x_j^{Min} je dolná hranica j-tého parametra.

3.3 Pertubácia

Mutácia je proces, pri ktorom dochádza k náhodnej zmene vlastností či parametrov daného jedinca. U SOMA algoritmu nejde o klasickú mutáciu, ale o tzv. pertubáciu. Pri pohybe jedincov priestorom možných riešení sa náhodne, za pomoci generovania náhodných čísel, ich pohyb ruší, čiže je pertubovaný. Sila pertubácie, a teda to, ako veľa parametrov sa jedincovi zmení, záleží na nastavení parametra *PRT*, ktorý si na začiatku volíme.

Podľa parametra *PRT* sa pre každého jedinca vygeneruje na začiatku každého skoku pertubačný vektor (*PRTVector*) s veľkosťou *D*. *PRTVector* je sekvencia náhodne vygenerovaných čísel 0 a 1. Pokiaľ je n-té vygenerované číslo väčšie ako parameter *PRT*, potom je parameter na n-tej pozícii *PRTVectoru* nastavený na 0, inak je jeho hodnota 1. Toto má vplyv na pohyb jedinca tak, že pokiaľ sú všetky hodnoty 1, tak aktuálny jedinec sa pohybuje rovno k *Leaderovi*, ak sú však niektoré prvky 0, rovnica sa vynuluje a daná súradnica sa nemení a ostáva táto hodnota rovnaká pre každý jeden skok jedinca. Tvorbu *PRT* môžeme popísať nasledujúcim spôsobom:

$$\text{if } (rnd_j < PRT) \text{ then } PRTVector_j = 1 \text{ else } 0 \ , j = 1, ..., N.$$

Tento proces sa opakuje pred každým novým skokom, čiže pohyb jedinca nevyzerá vždy rovnako, ale každým skokom sa mení. Šanca na nájdenie globálneho extrému sa zvyšuje, pretože zastavenie niektorých súradníc odkloní dráhu aktívneho jedinca od smeru vedúcemu k *Leaderovi*, tak sa prehľadáva aj tá časť možných riešení, ktoré by prehľadávané byť nemuseli.

3.4 Migrácia

Na začiatku migračného kola je každý jedinec ohodnotený účelovou funkciou. Taktiež sa na začiatku volí *Leader*, jedinec s najlepšou hodnotou (minimálnou alebo maximálnou), na nasledujúce migračné kolo. Ostatní jedinci sa začnú pohybovať smerom k zvolenému jedincovi *(Leaderovi)* pomocou skokov. Veľkosť skoku je daná parametrom *Step*. Po každom takomto prevedenom skoku si jedinec prepočíta svoju hodnotu účelovej funkcie, pokiaľ je lepšia, tak si ju zapamätá, pokiaľ nie, zapamätá si pôvodnú hodnotu. Pohyb jedinca po skokoch, ktoré smerujú k *Leaderovi* pokračuje, dokým nie je dosiahnutá pozícia daná parametrom *PathLength*.

Jedinec si pamätá pozíciu, kde bola nájdená najlepšia hodnota účelovej funkcie a po skončení behu sa na ňu presunie. Pohyb jedinca je daný vzťahom:

$$x_{i,j}^{MLnew} = x_{i,j,start}^{ML} + (x_{L,j}^{ML} - x_{i,j,start}^{ML}) \cdot t \cdot PRTVector_j,$$

kde $x_{i,j}^{MLnew}$ je nová pozícia jedinca,
$x_{i,j,start}^{ML}$ je pozícia aktívneho jedinca,
$x_{L,j}^{ML}$ je pozícia lídra,
ML značí aktuálne migračné kolo,
$t \in \langle 0,$ po *Step*, do *PathLength* $\rangle$.

3.5 Ukončovacia podmienka

Ukončenie algoritmu spočíva v tom, že sa kontroluje, či prebehli všetky migračné kolá nastavené v parametri *MigrationsCount*. Pokiaľ sa táto podmienka splní, algoritmus končí, pokiaľ nie, pokračuje ďalej. Taktiež je možné si zvoliť parameter *MinDiv* a pri ukončovaní algoritmu by sa kontrolovalo splnenie podmienky, že rozdiel medzi najlepším a najhorším jedincom nie je väčší ako tento parameter.

3.6 Princíp fungovania

V Algoritme 1 je znázornený SOMA pseudokód základnej stratégie „Všetci k jednému“ (AllToOne).

Algorithm 1 SOMA pseudokód

Vstupy:

počiatočná, náhodne vygenerovaná populácia

riadiace a ukončovacie parametre, ktoré sú uvedené v Tabuľke 1

f_{cost} - účelová funkcia, ktorá vracia vhodnosť aktuálneho riešenia

```
1: for i < MigrationsCount do
2:     výber lídra (najlepšieho jedinca)
3:     for j < PopSize do
4:         výber j-tého jedinca
5:         výpočet f_cost novej pozície jedinca
6:         ulož najlepšie nájdené riešenie do novej populácie
7:         if najlepsiJedinec − najhorsiJedinec > minDiv then
8:             stop SOMA
9:         end if
10:    end for
11: end for
```

3.7 Stratégie

Existuje niekoľko stratégií algoritmu SOMA. Všetky stratégie sú podobné a vymýšľanie nových stratégií je spôsobené tým, že všetky sa snažia čo najlepšie nájsť globálne optimum. Verzie stratégií sú následovné:

1. **AllToOne**: Prvá, základná stratégia je „Všetci k jednému“, z anglického „AllToOne“. Stratégia „AllToOne“ je prevedená tak, že sa všetci aktívni jedinci posúvajú za jedným, ktorý je označený ako líder. Líder je najlepší jedinec z populácie. Počas jedného migračného kola ostáva líder na svojej pozícii. Každým ďalším migračným kolom sa líder mení, takže nie je ten istý líder počas všetkých migračných kôl.

2. **AllToAll**: Stratégia „Všetci ku všetkým“, v ktorej neexistuje žiadny líder. Všetci jedinci migrujú za všetkými ostatnými tak ako pri „AllToOne“, s tým rozdielom, že aktuálny jedinec sa po skončení migračného kola vracia na pozíciu, kde našiel počas *PopSize*-1 migračných ciest najlepšiu hodnotu. Pravdepodobnosť na nájdenie globálnemu extrému sa zvyšuje, pretože jedinci takto prejdú väčší priestor možných riešení.

3. **AllToAll Adaptive**: Rozdiel medzi týmto algoritmom a „AllToAll“ je ten, že jedinec nezačína migráciu zo starej pozície (ako v prípade „AllToAll“), ale z posledne nájdenej najlepšej pozície, ktorú našiel počas posúvania k predchádzajúcim jedincom.

4. **AllToRand**: Stratégia, kde sa všetci jedinci posunú za náhodne zvoleným jedincom, ktorý sa mení v každom migračnom kole. Nezáleží na ohodnotení daného zvoleného jedinca, ale vyberá sa náhodne.

3.8 Modifikácie

Pôvodný SOMA algoritmus bol uvedený v roku 2004 Zelinkom [1]. Neskôr došlo k mnohým jeho vylepšeniam. Väčšinou ide o jeho úpravu či vylepšenie pôvodného algoritmu alebo ide o skombinovanie dvoch algoritmov, tzv. hybridné algoritmy, ktoré spolu vytvoria nové riešenia.

V roku 2007 predstavili Deep a Dipti samo-organizujúci sa migračný genetický algoritmus. Je to kombinácia migrácií a organizácie SOMA algoritmu spolu so selekciou, krížením a mutáciami z binárneho GA [17]. Tento algoritmus je prevedený tak, že každé migračné kolo nasleduje jedna generácia GA s krížením a bitovou mutáciou. Využíva teda výhody oboch algoritmov, SOMA i GA.

V roku 2009 bol uvedený Discrete Self-Organising Migrating Algorithm (DSOMA) [18]. Ide o tzv. diskrétnu verziu algoritmu, ktorá bola navrhnutá na riešenie permutačne založených kombinátorických optimalizačných problémov. DSOMA dokázal úspešne riešiť výrobné plány a problém zadeľovania úloh.

Kadlec a Raida v roku 2011 prezentovali novú variantu SOMA algoritmu, tzv. MOSOMA (Multi-Objective SOMA) [19]. V MOSOMA ide o to, že riešenia sú uložené vo frontách založených na dominantnosti riešenia, populácia skáče za všetkými jedincami v najlepšej fronte. Rieši problémy, kde je potrebné naraz optimalizovať viac účelových funkcií.

Neskôr v roku 2016 bol uvedený Self Organizing Migrating Algorithm with Quadratic Interpolation (SOMAQI) [20] a zaoberá sa riešením veľkej škály optimalizačných problémov dimenzií dosahujúcich hodnoty od 100 do 300 s konštantnou veľkosťou populácie. Prichádza s vysokou kvalitou optimálnych riešení s malou výpočtovou zložitosťou, taktiež veľmi rýchlo konverguje k optimálnemu riešeniu.

Skanderová a Zelinka uviedli v roku 2019 algoritmus SASOMA (Self-Adapting SOMA). Hlavnými znakmi nového algoritmu je to, že migračná stratégia je vyberaná podľa počtu jedincov, líder nepatrí do populácie, pertubačný vektor je generovaný na základe skupín premenných alebo parameter PRT je samo-adaptívny, ďalej parameter $Step$ je samo-adaptívny. Diverzita je zabezpečovaná nahradením jedincov použitím externých archívov [21].

V roku 2019 predstavili Tomaszek a Zelinka novú stratégiu nazývanú „AllToNBest“. Ide o kombináciu dvoch stratégií - „AllToOne“ a „AllToRand“. Pri výbere lídra sa jedinci z populácie zoradia podľa hodnoty účelovej funkcie a vyberie sa náhodný jedinec z N najlepších. Pokiaľ $N = 1$, SOMA pokračuje stratégiou „AllToOne“, pokiaľ bude $N = PopSize$, SOMA pokračuje stratégiou „AllToRand“. Ukázalo sa, že nová stratégia „AllToNBest“ dosahuje podobné alebo lepšie výsledky [22].

4 Umelé neuronové siete

Umelé neuronové siete sú jednoduché matematické modely, ktoré napodobujú činnosť biologických neurónových sietí, čiže ľudskú inteligenciu. Mali by byť schopné sa správať rovnako alebo aspoň podobne ako ich biologické vzory. Ich hlavnou schopnosťou je učenie sa na základe nejakých skúseností a prispôsobenie sa zmenám vo svojom okolí. Dnes sa umelé neurónové siete používajú pri rozpoznávaní vzorov [23], hlasu [24], písma [25], pri optimalizácii [26], pri kompresii dát [27] atď.

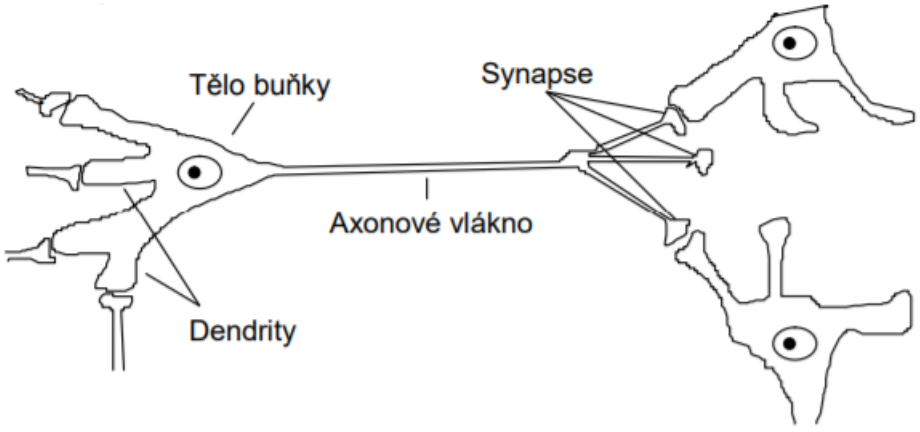
4.1 Neurón

Základná jednotka, ktorá má schopnosť šíriť a spracovávať signály sa volá neurón. Ľudský mozog má neurónov zhruba 10 miliárd [28]. Vchádza do nich veľa vstupov a po spracovaní bunkou sú premenené na výstup. Vstupná štruktúra sa skladá z dendritov a výstupná z tzv. axónov. Axón počiatočnej bunky sa pripojuje na dendrity ostatných buniek pomocou synapsií. Synaptický spoj je schopný prenášať signál.

Neurón reaguje na veľkosť podráždenia na svojich vstupoch a stane se aktívnym iba pokiaľ celkový súčet všetkých signálov obdržaných z dendritov prekročí istú úroveň (aktivačný prah). Túto prahovú hodnotu má telo každého neurónu. Po aktivácii neurónu posiela axón elektrický signál. Signál sa synapsiami šíri k ostatným neurónom, ktoré sa môžu taktiež aktivovať. Sila signálu závisí na sile synaptického spoja. Toto spojenie sa dá posilňovať učením.

Umelý neurón obsahuje niekoľko vstupov (dendrity), ktoré sú ohodnotené váhami, a jeden výstup (axón). V neuróne prebiehajú dva procesy – výpočet postsynaptického potenciálu a výpočet hodnoty výstupu pomocou aktivačnej funkcie. Tá je vo forme nelineárnej funkcie, najčastejšie sigmoidy. Postsynaptický potenciál je rovný súčtu všetkých vstupných signálov násobených príslušnými váhami. Pokiaľ prekročí hodnota postsynaptického potenciálu prahovú hodnotu, neurón sa excituje.

Zjednodušene biologický neurón je znázornený na Obr. 2.



Obr. 2: Biologický model neurónu [2]

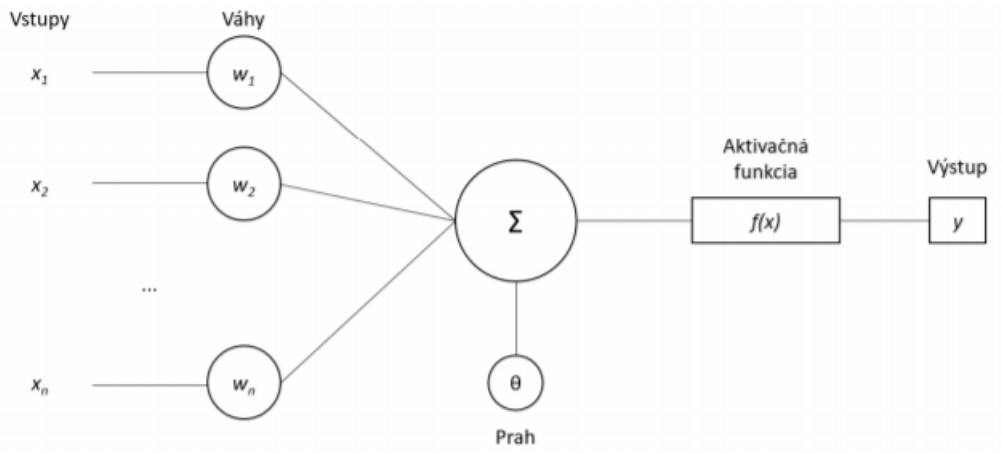
Jednotlivé body popisujúce model neurónu:

1. Dendrity reprezentujú miesto vstupu signálov do tela neurónu.

2. Telo bunky sčítava signály dané okolitými neurónmi. Takto stanovený vnútorný potenciál vedie k excitácii (vybudeniu) neurónu.

3. Axonové vlákno prenáša signál daný stupňom excitácie k synapsiam.

4. Synapsia tvorí výstupné zariadenie neurónov, ktoré signál zosilňujú alebo zoslabujú a predávajú ďalším neurónom.



Obr. 3: Fungovanie neurónu [2]

Na Obr. 3 je znázornené fungovanie neurónu. Znak x_i pre $i = 1, 2, ..., N$ predstavuje prichádzajúce signály od ostatných neurónov. Sila synaptického spoja je daná váhami w_i. Súčet vážených signálov tvorí postsynaptický potenciál Σ. Ten vstupuje ako argument do aktivačnej funkcie f, ktorá vráti výstup y.

Každý neurón má N vstupov a jeden výstup. Transformácia vstupov na výstup prebieha podľa vzťahu:

$$Y = f(\sum_{n=1}^{N} w_i \cdot x_i + \theta),$$

kde f je prechodová funkcia neurónu
w_i je váha i-tej vstupnej synapsie neurónu,
x_i je i-tý vstup neurónu,
θ označuje prah neurónu.

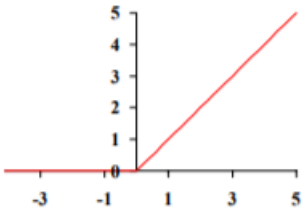
4.2 Prenosové a aktivačné funkcie

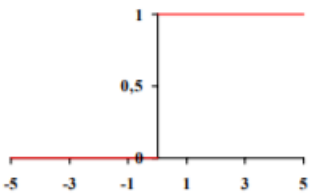
Popíšeme si základné aktivačné funkcie neurónových sietí. Hoci je týchto funkcií veľké množstvo, vyberieme si najpoužívanejšie z nich. Medzi hlavných predstaviteľov by sme mohli zaradiť

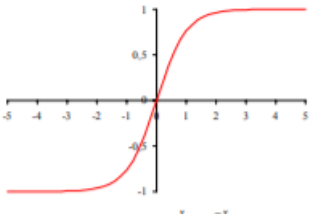
sigmoidálnu prechodovú funkciu, binárnu či ReLU funkciu, ktoré si nižšie popíšeme. Znázornené sú na Obr. 4.

a) ReLU - rieši iba lineárne separabilné problémy.
$f(P) = x \ \forall x > 0$, inak $x = 0$.
b) Binárna - skoková funkcia nadobúdajúca iba hodnoty 0 a 1.
$f(B) = 1 \ \forall x > 0$, inak $x = 0$.
c) Logistická (sigmoidálna) - najpoužívanejšia prenosová funkcia odvodená ako aproximácia prenosovej funkcie biologického neurónu

$$f(S) = \frac{1}{1 + e^{-\lambda}}$$

a) 

b) 

c) 

Obr. 4: Typy prenosových funkcií [29]

4.3 Typy neurónových sietí

Umelé neurónové siete sú univerzálne - dokážu uskutočňovať transformáciu nad vstupnými dátami. Sú robustné - informácie sú rozšírené po celej sieti, obsahujú veľa neurónov a synapsií. Dokážu reagovať na iné vstupné dáta než len na tie, ktoré boli použité v trénovacej množine, čiže sú taktiež schopné generalizácie a abstrakcie.

Je ich možné rozdeliť na nerekurentné - dopredné a rekurentné. Dopredné na rovnaký vstupný stimul vrátia vždy rovnakú odozvu siete. Informačný tok je jednoznačne definovaný, čiže neexistujú žiadne prepojenia v rámci rovnakej vrstvy ani prepojenia z vyšších vrstiev do nižších. Naopak odozva rekurentnej siete nie je daná, na rozdiel od doprednej siete, iba vstupným stimulom, ale taktiež odráža vplyv stimulov, ktoré predchádzali aktuálnemu stimulu. V sieti je teda potrebné implementovať aj časový kontext za pomoci rekurentných neurónov.

4.3.1 Dopredné siete

Dopredné neurónové siete sú neurónové siete, kde spojenie medzi dvoma jednotkami netvorí cyklus. Dopredné siete boli prvý typ umelých neurónových sietí, sú jednoduchšie než siete rekurentné. Ich názov je odvodený od faktu, že informácia prejde dopredu sieťou a netvoria sa pritom žiadne cykly. Najskôr prejde informácia vstupnou vrstvou, potom skrytými vrstvami (pokiaľ tam nejaké sú) a následne výstupnými uzlami [30].

Dopredné siete sa využívajú najmä na učenie v prípadoch, kde učené dáta nie sú ani sekvenčné ani časovo-závislé.

4.3.2 Perceptron

Najjednoduchší typ doprednej siete je perceptron. Perceptron je jednoduchá neurónová sieť bez skrytých vrstiev. Je to prvá umelá sieť, ktorá bola schopná učenia. Má iba vstupnú a výstupnú vrstvu. Výstupy sú počítané priamo zo súčtu výsledkov násobenia jednotlivých váh a vstupov. Prvá vrstva tejto siete nemá meniteľné váhy. Váhy druhej vrstvy sú nastaviteľné. Perceptron sa používa na rozdelenie priestoru do dvoch tried. Výstup je binárny, čiže klasifikuje buď do triedy 0 alebo 1. Prenosová funkcia perceptronu je potom skoková [31].

4.3.3 Viacvrstvový perceptron

Viacvrstvový perceptron je umelá neurónová sieť, ktorá sa skladá z viacerých perceptronov. Narozdiel od jednoduchého perceptronu, je schopný sa učiť počítať nelineárne separabilné funkcie. Vďaka ich schopnosti učiť sa nelineárne funkcie sa využívajú na učenie klasifikácie aj regresie.

Obvykle sú viacvrstvové perceptrony organizované do vrstiev. Pozostáva zo vstupnej vrstvy, skrytých vrstiev a výstupnej vrstvy. Narozdiel od jednoduchého perceptronu má minimálne jednu skrytú vrstvu.

Každý perceptron na vrstve l_i je spojený s perceptronom na vrstve l_{i-1}. Vrstvy sú plne prepojené. Každý perceptron závisí od výstupov všetkých perceptronov z predchádzajúcej vrstvy. Perceptrony na rovnakej vrstve nie sú medzi sebou prepojené [31].

4.3.4 Rekurentné siete

Rekurentné siete obsahujú minimálne jednu cyklickú cestu, kde rovnaká informácia na vstupe ovplyvňuje opakovane aktivitu neurónov na tejto ceste. Informácia prechádza v cykloch z vrstvy do vrstvy. Aktuálnu vrstvu ovplyvňuje predchádzajúca vrstva. Rekurentné siete majú pamäť, ktorá im umožňuje uchovávať informácie z predchádzajúcich výpočtov. Rovnako ako dopredné siete majú skryté vrstvy. Narozdiel od dopredných sietí majú vrstvy prepojenie na seba, čo umožňuje výsledky skrytej vrstvy v jednej časovej inštancii použiť ako vstup skrytej vrstvy v ďalšej časovej inštancii. Toto vytvára pamäť, ktorá, pokiaľ je trénovaná, dovoľuje skrytým vrstvám zbierať informácie o dočasnom vzťahu medzi vstupnou a výstupnou sekvenciou [32].

Rekurentné sú preto, lebo uskutočňujú rovnaké výpočty pre všetky elementy vstupnej sekvencie. Rozdiel medzi výstupmi rôznych elementov zo vstupnej sekvencie vychádza z rozdielu skrytých vrstiev, ktoré sú závislé na aktuálnom elemente na vstupnej sekvencii a hodnote skrytej vrstvy v poslednom kroku. Siete majú bližšie k biologickým neurónovým sieťam.

Využívajú sa na asociačné a predikčné úlohy s časovým kontextom [33] či pri modelovaní sekvencií [34].

4.3.5 Konvolučné siete

Konvolučné neurónové siete sú siete, ktoré pozostávajú z jednej alebo viac konvolučných vrstiev, ktoré sú nasledované jednou alebo viacerými plne prepojenými vrstvami ako pri štandardných viacvrstvových sieťach. Sú navrhnuté k využívaniu 2D štruktúry vstupných obrázkov. Výhodou takýchto sietí je to, že sa jednoducho učia a majú oveľa menej parametrov ako plne prepojené siete s rovnakým počtom skrytých vrstiev.

Hoci boli špeciálne navrhnuté na rozpoznávanie 2D tvarov, ukázalo sa ale, že je možné ich využívať vo viacerých odvetviach. Dajú sa využiť pri rozpoznávaní ľudskej tváre [35], pri diagnostikovaní Alzheimerovej choroby [36], ale aj pri klasifikácii časových sekvencií [37] či pri detekovaní a lokalizácii zemetrasení [38].

5 Analýza, návrh a vývoj

V tejto kapitole si predstavíme konkrétne prevedenie a návrh nového algoritmu, ukážeme si užívateľské rozhranie a rozoberieme si hodnoty jednotlivých výsledkov testov, ktoré dosiahli oba algoritmy - pôvodný SOMA algoritmus, ale taktiež algoritmus SOMANN a porovnáme si ich výsledné hodnoty z testovacích funkcií. Prestavíme si aj jednotlivé kroky na implementáciu SOMANN algoritmu.

5.1 Požiadavky

Vylepšiť pôvodný SOMA algoritmus tak, aby vznikla nová varianta algoritmu, prípadne lepšia a rýchlejšia. Ďalšie vylepšenie by mohlo byť dosiahnuté využitím adaptačných parametrov. Úlohou bude teda overiť, či takéto riešenie môžeme dosiahnuť za pomoci neurónových sietí. Vstupom tejto siete môžu byť informácie o populácii - rozmiestnenie jedincov, vzdialenosť od lídra atď. Výstupom danej siete budú jednotlivé parametre SOMA algoritmu, upravené neurónovou sieťou.

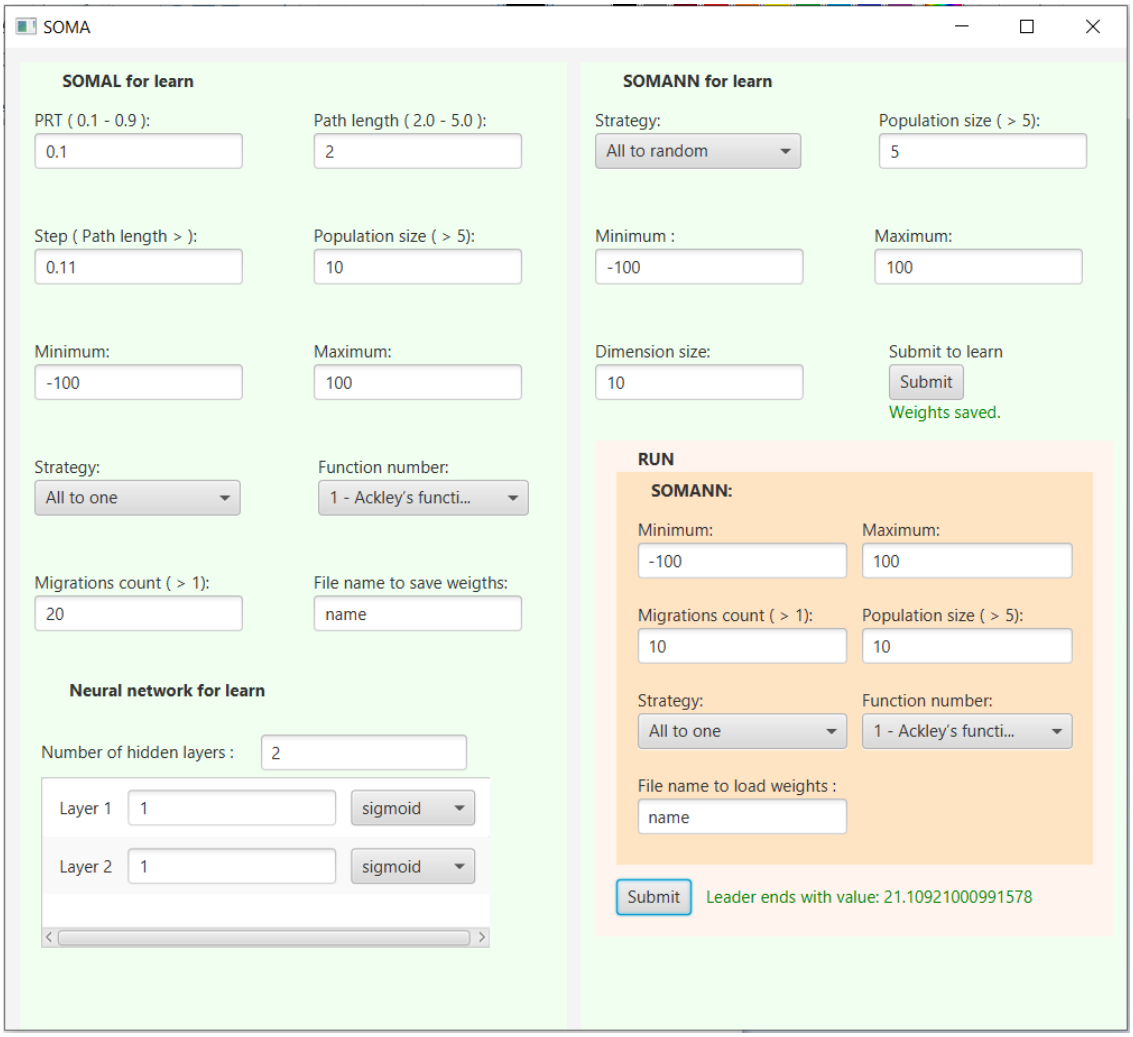
5.2 Voľba vývojového nástroja

Dané riešenie je implementované v jazyku Java. Tento jazyk som si vybrala preto, lebo Java je kompilovaný jazyk, čiže sa najskôr zdrojový kód preloží prekladačom do strojového kódu. V porovnaní napríklad s jazykom Python by mala byť Java v tomto zmysle rýchlejšia, pretože Python je jazyk interpretovaný, čiže taký, ktorý potrebuje k spusteniu nejaký ďalší program zvaný interpret, ktorý zdrojový kód tohto jazyka interpretuje, čím sa stáva pomalším.

5.3 Návrh užívateľského rozhrania

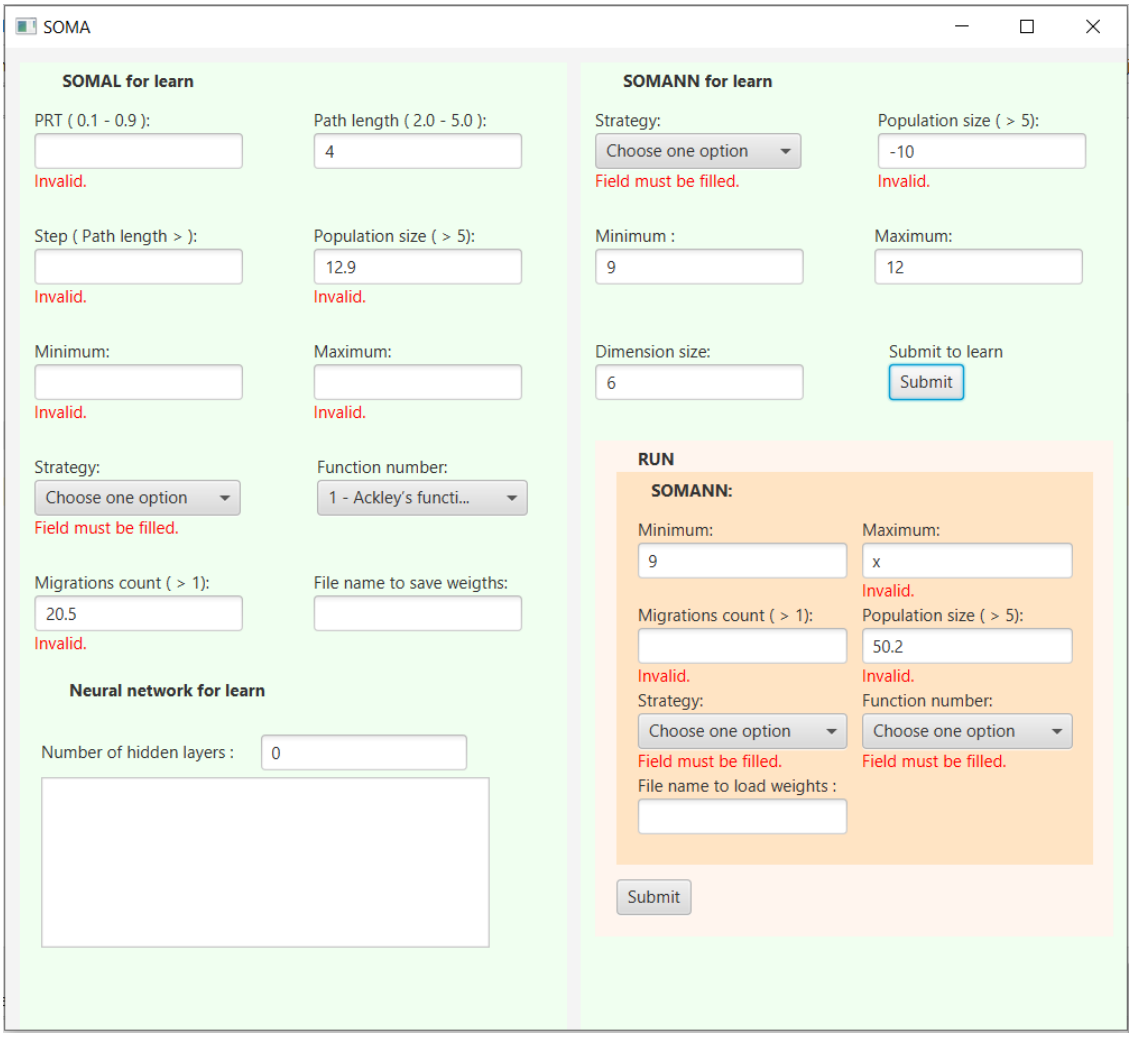
Na Obr. 5 je znázornené prevedenie grafického užívateľského rozhrania, kde si môže užívateľ zvoliť počiatočné parametre pre trénovanie vlastnej siete. Pre učenie siete slúži zelená časť - v prvej časti si vyberáme jednotlivé parametre pre SOMA_L algoritmus, ďalej si volíme meno súboru, do ktorého sa majú uložiť váhy siete a informácie o sieti. V ďalšej časti si volíme skryté vrstvy siete, vstupná a výstupná sú dané, na skrytých si môžeme zvoliť aktivačnú funkciu a počet neurónov na jednej vrstve. Ďalej sa volia parametre pre SOMANN algoritmus. Na konci sa uložia váhy siete spolu so skrytými vrstvami do súboru.

Zo súboru s rovnakým názvom potom tieto váhy a informácie o skrytých vrstvách môžeme načítať a budú nastavené sieti, pokiaľ nezadáme meno súboru alebo zlý súbor, je vybraná sieť, ktorá bola použitá pri učení a spúšťaní testovacích funkcií. V oranžovej časti (run SOMANN) pre spúšťanie algoritmu si potom volíme zvyšné parametre, ktoré chceme, aby do algoritmu vstupovali. Po skončení behu sa vypíše hodnota lídra, ktorého sme dostali.



Obr. 5: Grafické užívateľské rozhranie

Ošetrenie zvolených vstupov je znázornené na Obr. 6. Po zadaní nesprávnych vstupov alebo nezadaní potrebného vstupu sa vypíše chybová hláška.



Obr. 6: Grafické užívateľské rozhranie - chybové hlášky

6 Nový SOMA algoritmus s neurónovou sieťou

Na úvod si predstavíme nový algoritmus SOMANN ako rozšírenie pôvodného algoritmu o neurónovú sieť, ktorej úlohou je adaptácia parametrov SOMA algoritmu. Neurónová sieť rozhoduje podľa zadaných vstupov o tom, ako budú pre každého jedinca zvlásť nastavené parametre, ktoré dostaneme ako výstup zo siete. V novom algoritme ide o zefektívnenie pôvodného algoritmu neurónovou sieťou a dosiahnutie čo najlepších výsledkov optimalizáciou pôvodných parametrov.

Prvým krokom je voľba parametrov pre algoritmus SOMANN, tie si volí užívateľ hneď na začiatku, ako je uvedené v predchádzajúcej kapitole. Následne je daný algoritmus prevedený tak, že z vygenerovanej populácie vyberieme „aktuálneho“ jedinca, ktorému sa vypočítajú na začiatku zvolené vlastnosti, tieto jeho vlastnosti vstupujú do neurónovej siete, počet neurónov na vstupnej vrstve sa rovná počtu nami zvolených vlastností jedinca. Jedná sa o vzdialenosť od lídra, diverzita, normované id jedinca, rozdiel vhodností najlepšieho a aktuálneho jedinca, rozdiel vhodností priemerného a aktuálneho jedinca. Výstupy siete sú parametre SOMANN algoritmu, konkrétne nastavenie pre „aktuálneho“ jedinca počas migračného kola. Použitá neurónová sieť je dopredná, prechodová funkcia siete je sigmoid.

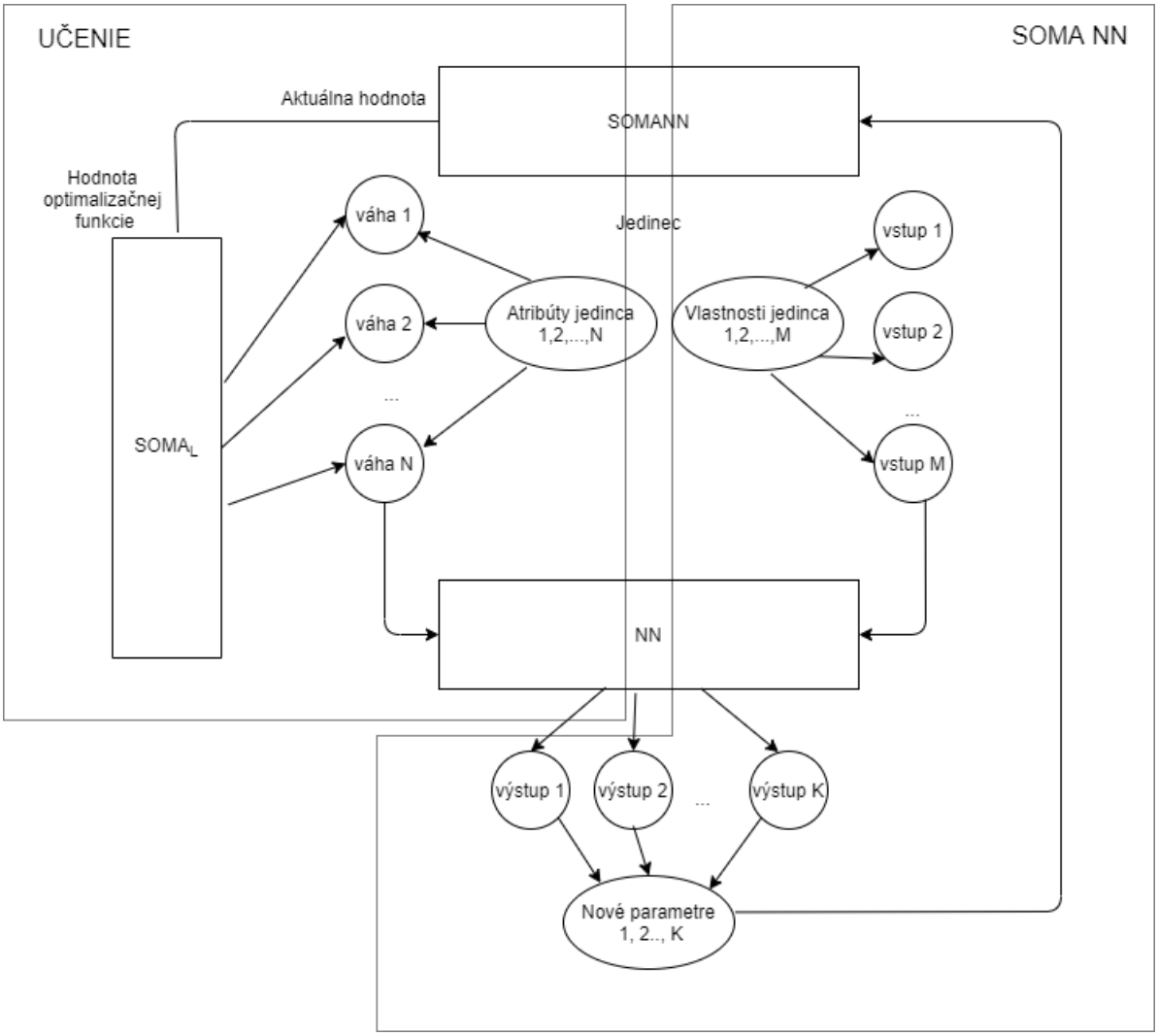
6.1 Učenie

O učenie neurónovej siete sa stará algoritmus SOMA$_L$. Hľadá čo najlepšie váhy pre neurónovú sieť. Použitá sieť je dopredná, má 4 vrstvy, na vstupnej vrstve má 5 vstupov, na skrytej vrstve má 2 neuróny a na výstupnej vrstve má 3 výstupy.

Algoritmus SOMA$_L$ je ohodnocovaný účelovou funkciou. Vybraný je aktuálny jedinec, ktorého jednotlivé atribúty sú prevedené na váhy tejto siete. Prevedenie jedinca na váhy je popísané v 6.1.1. O sieti, ktorá vznikne priradením jedinca na váhy, ešte nevieme povedať ako správna alebo vhodná bude. Preto je táto neurónová sieť spustená 5 behov (opakovaní) na algoritme SOMANN. Po skončení 5 behoch je vybraný líder, čiže jedinec z najlepšou hodnotou vhodnosti. Súčet vybraných najlepších jedincov po 5 behoch je použitý ako hodnota účelovej funkcie. Týmto spôsobom sa jednotlivo testuje niekoľko sietí, na konci sa vyberie najlepšia sieť, čiže sieť s najlepšími váhami, ktoré sú nastavené na atribúty lídra z predošlých behov. Atribúty najlepšieho jedinca, ktorého dostaneme a čiže najlepšie nájdené váhy sa potom dajú uložiť do súboru, ktorý si môžeme zvoliť.

Po skončení učenia, keď je už sieť natrénovaná, načítame zo súboru najlepšie nájdené váhy a nastavíme ich sieti na začiatku nového behu. Naučená sieť potom rozhoduje v novom algoritme o tom, ako budú nastavené parametre vstupujúce do SOMANN algoritmu jednotlivo pre každého jedinca v migračnom kole. Ten s týmto nastavením urobí jedno migračné kolo, po ktorom je následne priradený do populácie.

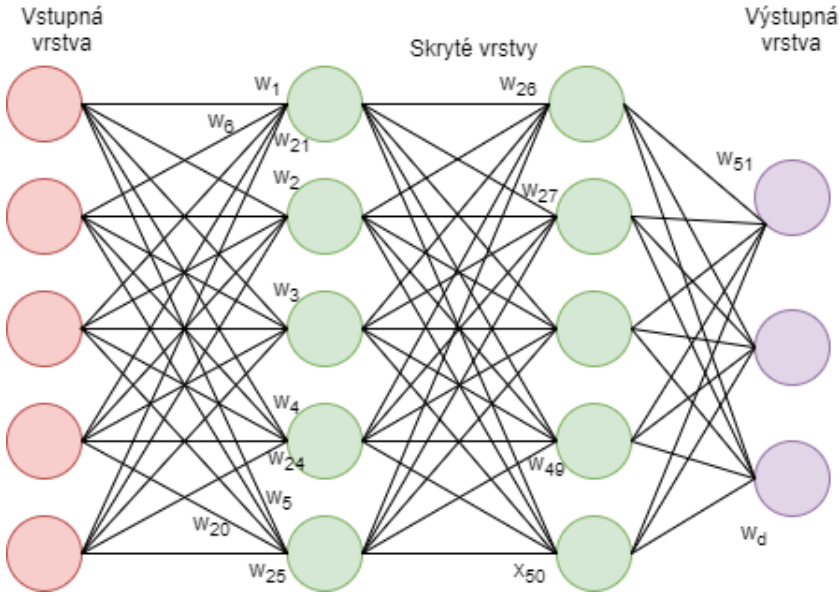
Fungovanie algoritmu je znázornené na Obr. 7.



Obr. 7: Fungovanie algoritmu SOMANN

6.1.1 Tvorba neurónovej siete

Váhy siete použité pri učení predstavuje vybraný jedinec. Tento jedinec SOMA$_L$ s dimenziou D a atribútmi $w_1, w_2, ..., w_d$ sa priraďuje do neurónovej siete po každom jeho atribúte, prechádza sa teda v cykle a priraďovanie prebieha po jeho jednotlivých atribútoch, začína sa od prvej vrstvy prvého neurónu, pokračuje sa prvou vrstvou druhého neurónu. Počet váh pre jednotlivé vrstvy sa rovná dimenzii jedinca. Priradenie jedinca do neurónovej siete je znázornené na Obr. 8.



Obr. 8: Priradenie jedinca do neurónovej siete

6.1.2 Účelová funkcia

Účelová funkcia sa snaží stanoviť extrém, hľadá ho na jednej z testovacích funkcií (napr. Rastriginova alebo Ackleyho). Ohodnocovanie účelovou funkciou je potrebné na to, aby sme dokázali nájsť čo najlepšieho jedinca, ktorého použijeme neskôr ako nastavenie pre váhy siete. SOMA$_L$ za ohodnotenia účelovej funkcie slúži aj na prenastavenie váh, tým že je ohodnocovaný jedinec a hľadá sa ten najlepší, hľadajú sa aj najlepšie váhy pre sieť. Účelová funkcia je súčet hodnôt vhodností lídrov SOMA$_L$ algoritmu po 5 behoch SOMANN. Na konci potom dostaneme lídra a nastavíme ho ako váhy sieti, sieť si uložíme následne do súboru.

6.2 Vstupy neurónovej siete

Vstupmi pre neurónovú sieť sú vlastnosti aktuálneho jedinca a populácie. Počet vstupov do neurónovej siete závisí od počtu vlastností, ktoré jedincovi počítame. Pri počte $k_1, k2, ...k_M$ vlastností je počet vstupov na vstupnej vrstve M. Váhy neurónovej siete sú atribúty jedinca, čiže parametre jeho polohy, čo pre jedinca s atribútmi $x_1, x_2, x_3, ..., x_N$ znamená, že N je počet váh neurónovej siete.

6.2.1 Výpočet vlastností

Pre populáciu s veľkosťou *PopSize* zisťujeme normované id aktuálneho jedinca z tejto populácie podľa vzťahu:

$$k_1 = \frac{i}{PopSize},$$

kde i je i-tý index.

PopSize označuje veľkosť populácie.

Vzdialenosť od lídra sa počíta tak, že pre jedinca s atribútmi $x_{i,1}, x_{i,2}, ..x_{i,N}$ a pre druhého jedinca s atribútmi $x_{j,1}, x_{j,2}, ..x_{j,N}$ je vzťah pre vzdialenosť $d_{i,j}$ týchto dvoch jedincov:

$$d_{i,j} = \sqrt{\sum_{k=1}^{N}(x_{i,k} - x_{j,k})^2}.$$

Diverzitu počítame podľa vzťahu [39]:

$$D = \frac{1}{PopSize} \sum_{i=1}^{PopSize} (\frac{1}{PopSize} \sum_{j=1}^{PopSize} d_{i,j}),$$

kde *PopSize* je veľkosť populácie.

Rozdiel hodnôt vhodnosti priemerného a aktuálneho jedinca je vypočítaný tak, že aktuálny jedinec s vhodnosťou f_i a priemerná vhodnosť celej populácie s veľkosťou *PopSize* sa počíta podľa vzťahu:

$$k_4 = f(x_i) - \sum_{j=1}^{PopSize} f(x_j)/PopSize,$$

kde $f(x_i)$ značí vhodnosť aktuálneho jedinca,

x_j je j-tý jedinec populácie.

Rozdiel hodnôt vhodnosti najlepšieho a aktuálneho jedinca znamená, že vhodnosť aktuálneho jedinca s hodnotou $f(x_i)$ odčítame od vhodnosti jedinca s najlepšou hodnotou vhodnosti $f(x_L)$, podľa vzťahu:

$$k_5 = f(x_i) - f(x_L).$$

6.2.2 Výstupy neurónovej siete

Výstupy neurónovej siete sú parametre, ktoré potom vstupujú do SOMANN algoritmu. Tieto parametre sú zvolené iba počas jednej migrácie aktuálneho jedinca. Daný jedinec urobí s týmto nastavením migráciu, po ktorej vznikne jeho nová poloha. Parametre, ktoré sa nastavujú jedincovi pre jedno migračné kolo sú *PRT*, *PathLength* a *Step*. Po získaní výstupov zo siete je potrebné ich upraviť, aby sa pohybovali v zvolenom rozmedzí, keďže je použitá sigmoid funkcia, ktorej hodnoty veľakrát nedosahujú ani hodnoty 0,1. Na docielenie toho, že parametre budú nadobúdať hodnoty z celého rozmedzia odporúčaných hodnôt pre konkrétny parameter, je potrebné ich upraviť.

Parameter *PRT* má odporúčané hodnoty od 0,1 do 0,9. Hodnotu, ktorú vráti neurónová sieť PRT_{nn}, je potrebné upraviť podľa vzťahu:

$PRT = 0,9 \cdot PRT_{nn} + 0,1.$

Hodnoty *PathLength* by sa mali pohybovať v hodnotách od 2 do 5 a preto hodnota $PathLength_{nn}$, ktorú vráti sieť, upravíme pomocou vzťahu:

$PathLength = 4 \cdot PathLength_{nn} + 1.$

Ďalej bolo potrebné upraviť aj hodnotu $Step_{nn}$, ktorý nám vráti sieť, rovnako podľa vzťahu: $Step = 0,9 \cdot Step_{nn} + 0,1$, aby sme dostali hodnoty od 0,1 do 0,9.

7 Testovanie

V tejto sekcii si ukážeme výsledky testovania 2 algoritmov - SOMA a SOMANN. Popíšeme si jednotlivé testovacie funkcie a porovnáme si výsledné hodnoty na oboch algoritmov. Nasleduje popis funkcií, na ktorých boli prevedené experimenty.

7.1 Testovacie funkcie

a) Ackleyho funkcia:

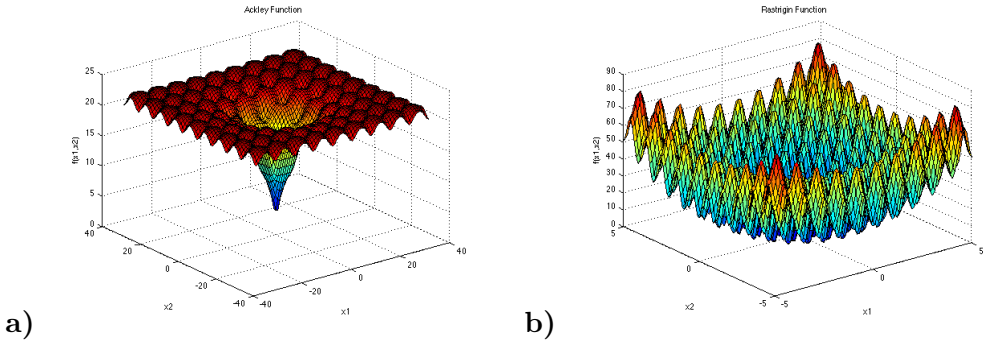
$$f(x_0 \cdots x_n) = -20exp(-0,2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + e.$$

-32 $\leq x_i \leq 32$, minimum v $f(0, \cdots, 0) = 0$.
Ackleyho funkcia je zobrazená na Obr. 9.

b) Rastirginova funkcia:

$$f(x_1 \cdots x_n) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i))$$

-5,12 $\leq x_i \leq 5,12$, minimum v $f(0, \cdots, 0) = 0$.
Rastirginova funkcia je zobrazená na Obr. 9.



a) **b)**

Obr. 9: Ackleyho a Rastirginova funkcia [40]

c) De Jongova funkcia č.1:

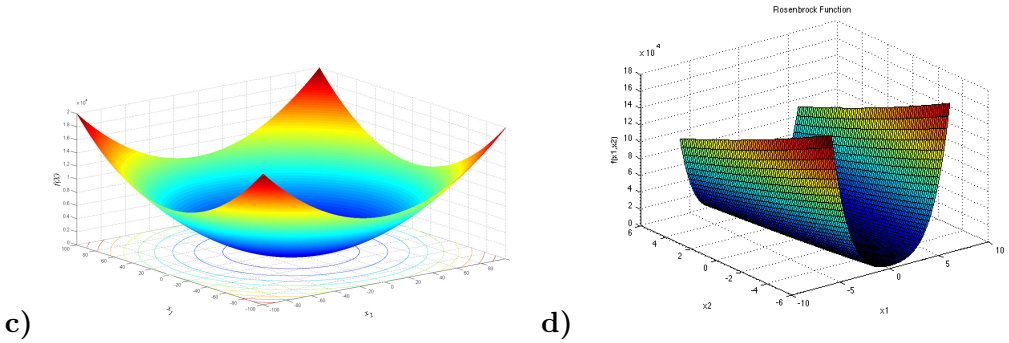
$$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{i=1}^{n}(x_i^2).$$

-5,12 $\leq x_i \leq 5,12$, minimum v $f(0, \cdots, 0) = 0$.
De Jongova funkcia č.1 je zobrazená na Obr. 10.

d) Rosenbrockova funkcia:

$$f(x_1 \cdots x_n) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2).$$

-2,048 $\leq x_i \leq 2,048$, minimum v $f(1, 1, \cdots, 1) = 0$.
Rosenbrockova funkcia je zobrazená na Obr. 10.



c) **d)**

Obr. 10: De Jongova [41] a Rosenbrockova funkcia [40]

e) Schwefelova funkcia:

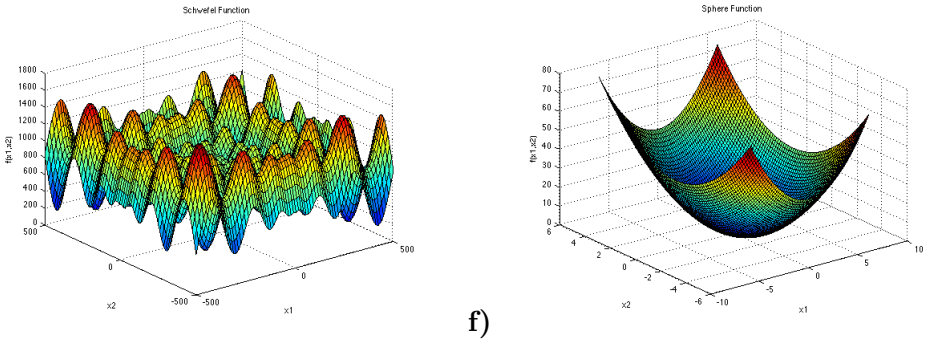
$$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = 418,9829d - \sum_{i=1}^{n} x_i sin(\sqrt{|x_i|}).$$

-512 $\leq x_i \leq 512$, minimum v $f(420,968746, 420,968746, \cdots, 420,968746) = 0$.
Schwefelova funkcia je zobrazená na Obr. 11.

f) Sférická funkcia:

$$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i^2.$$

-5,12 $\leq x_i \leq 5,12$, minimum v $f(0, \cdots, 0) = 0$.
Sférická funkcia je zobrazená na Obr. 11.



e) **f)**

Obr. 11: Schwefelova a Sférická funkcia [40]

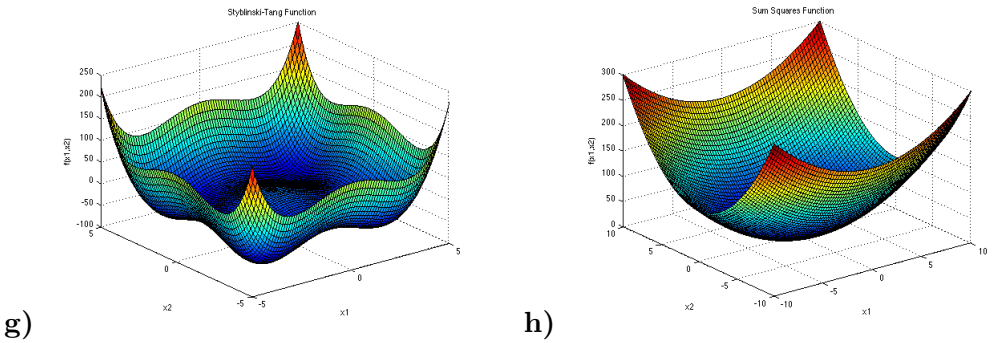
g) Styblinski-Tang funkcia:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = \frac{1}{2}\sum_{i=1}^{n}(x_i^4 - 16x_i^2 + 5x_i).$$

-5 $\leq x_i \leq 5$, minimum v $f(-2,903534, \cdots, -2,903534) = 39,16599$.
Styblinksi-Tang funkcia je zobrazená na Obr. 12.
h) Sum Squares funkcia:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{i=1}^{n} ix_i^2.$$

-10 $\leq x_i \leq 10$, minimum v $f(0, \cdots, 0) = 0$.
Sum Squares funkcia je zobrazená na Obr. 12.



g) **h)**

Obr. 12: Styblinksi-Tang a Sum Squares funkcie [40]

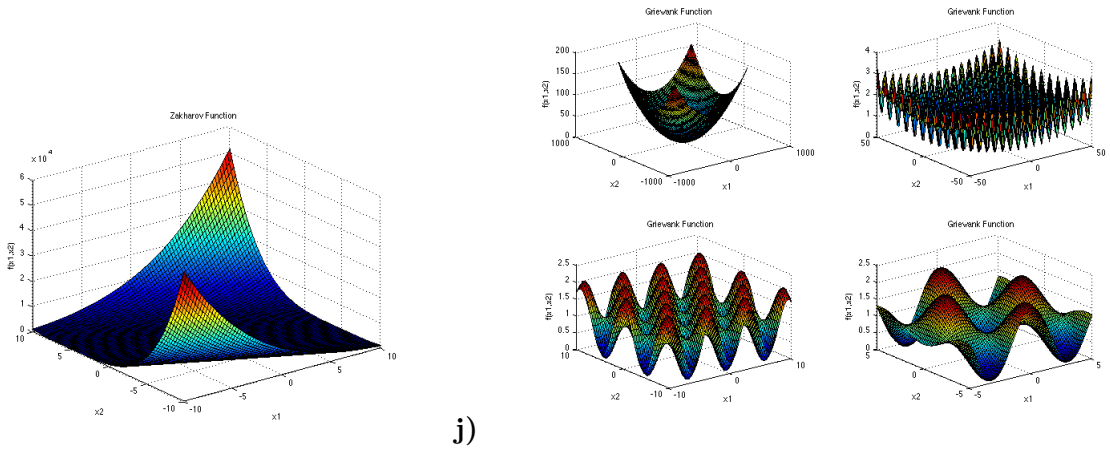
i) Zakharova funkcia:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} .0.5ix_i)^4.$$

-5 $\leq x_i \leq 10$, minimum v $f(0, \cdots, 0) = 0$.
Zakharova funkcia je zobrazená na Obr. 13.
j) Griewankova funkcia:

$$f(\mathbf{x}) = f(x_1, ..., x_n) = 1 + \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} cos(\frac{x_i}{\sqrt{i}}).$$

-512 $\leq x_i \leq 512$, minimum v $f(0, \cdots, 0) = 0$.
Griewankova funkcia je zobrazená na Obr. 13.



i) j)

Obr. 13: Zakharova a Griewankova funkcia [40]

7.2 Výsledky testovania na algoritmoch SOMA a SOMANN

V oboch prípadoch pre SOMA i SOMANN bola použitá rovnaká stratégia, keďže išlo o porovnávanie na jednotlivých testovacích funkciách. Zvolená stratégia bola pre oba algoritmy „Všetci k jednému“ (AllToOne). Testovanie prebiehalo na všetkých 10 zvolených testovacích funkciách. Jednotlivé algoritmy vykonali 31 behov s rovnakým počtom ohodnotení na každej funkcii. Parametre, ktoré som zvolila pre SOMA algoritmus sú v Tabuľke 2.

Parametre použité pri testovaní sú pre SOMA$_L$ uvedené v Tabuľke 3. Parametre pre SOMANN sú uvedené v Tabuľke 4.

PRT	**Step**	**PathLength**	**PopSize**	**dimenzia**	**min**	**max**	**počet ohodnotení**
0,2	0,11	3,1	20	10	-100	100	35000

Tabuľka 2: Parametre SOMA algoritmu

PRT	**Step**	**PathLength**	**PopSize**	**min**	**max**	**počet ohodnotení**
0,2	0,11	3,1	20	-1	1	35000

Tabuľka 3: Parametre použité pre SOMA$_L$

PopSize	**min**	**max**	**dimenzia**
10	-100	100	10

Tabuľka 4: Parametre použité pre SOMANN

Algoritmus	Minimum	Maximum	Priemer	Medián	Smerodajná odchýlka
Ackleyho funkcia					
SOMA	2,00E+01	2,00E+01	2,00E+01	2,00E+01	1,15E-06
SOMA + NN	2,00E+01	2,00E+01	2,00E+01	2,00E+01	1,42E-14
Rastirginova funkcia					
SOMA	3,11E+00	1,93E+01	6,12E+00	6,80E+00	2,78E+00
SOMA + NN	2,07E+01	2,24E+02	9,65E+01	7,65E+01	5,23E+01
Rosenbrockova funkcia					
SOMA	1,73E+01	1,60E+02	7,62E+01	8,78E+01	3,91E+01
SOMA + NN	**8,96E-01**	3,08E+02	**6,15E+01**	1,85E+01	8,05E+01
De Jongova funkcia č. 1					
SOMA	1,73E-05	1,34E-04	6,82E-05	6,43E-05	2,74E-05
SOMA + NN	**6,28E-25**	**4,53E-17**	**1,85E-18**	**3,25E-20**	8,07E-18
Schwefelova funkcia					
SOMA	-1,34E+03	-1,25E+03	-1,30E+03	-1,31E+03	1,86E+01
SOMA + NN	-1,25E+03	-1,12E+03	-1,20E+03	-1,21E+03	3,16E+01
Sférická funkcia					
SOMA	2,42E-05	2,34E-04	9,53E-05	7,38E-05	5,25E-05
SOMA + NN	**3,64E-23**	**1,34E-16**	**5,04E-18**	**8,16E-21**	2,37E-17
Styblinksi-Tang funkcia					
SOMA	-8,22E+02	-7,66E+02	-7,96E+02	-7,94E+02	1,94E+01
SOMA + NN	-7,66E+02	-6,39E+02	-7,08E+02	-7,09E+02	3,36E+01
Griewankova funkcia					
SOMA	2,93E-06	1,23E-02	1,38E-03	2,45E-05	3,18E-03
SOMA + NN	**2,22E-16**	1,75E-01	3,79E-02	2,46E-02	4,16E-02
Sum Squares funkcia					
SOMA	1,11E-04	7,91E-04	3,14E-04	2,92E-04	1,44E-04
SOMA + NN	**8,45E-27**	**1,97E-16**	**7,96E-18**	**1,57E-22**	3,51E-17
Zakharova funkcia					
SOMA	9,52E-03	1,63E-01	5,15E-02	3,89E-02	3,64E-02
SOMA + NN	**1,74E-21**	**1,07E-12**	**7,54E-14**	**4,20E-15**	1,99E-13

Tabuľka 5: Porovnanie výsledných hodnôt algoritmov na funkciách SOMA a SOMANN

7.3 Porovnanie výsledkov na oboch algoritmoch

Porovnávala som 2 algoritmy - pôvodný SOMA algoritmus a algoritmus SOMANN. Výsledné hodnoty z oboch testovaní sú v Tabuľke 5.

Algoritmus SOMANN dokázal nájsť z 10 funkcií 6-krát celkovú minimálnu hodnotu, raz boli výsledky dvoch minimálnych hodnôt rovnaké, v ďalších dvoch prípadoch, kde algoritmy mali záporné hodnoty, boli výsledky porovnateľné, ale nižšie dosiahol SOMA, v jednom prípade algoritmus SOMANN dosiahol podstatnejšie vyššie hodnoty minima než pôvodný algoritmus.

V 4 prípadoch nadobudol nový algoritmus nižšie maximum ako originálny algoritmus, v jednom prípade boli hodnoty maxima rovnaké. Zvyšné hodnoty maxima na oboch algoritmoch boli porovnateľné, ale pôvodný algoritmus dosahoval menších hodnôt.

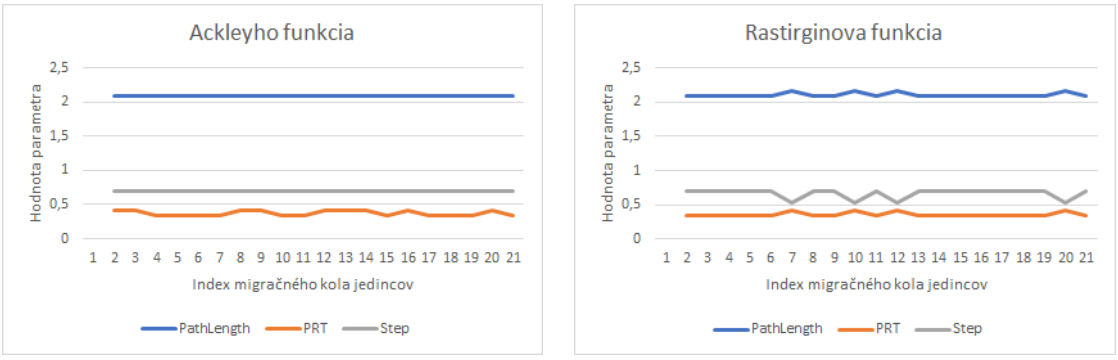
Až v polovici prípadov mal SOMANN algoritmus lepšie hodnoty priemeru než pôvodný algoritmus. V jednom prípade opäť dosahovali oba algoritmy rovnaké priemerné hodnoty. Na ďalších funkciách boli výsledky opäť porovnateľné a originálny algoritmus dosiahol tesne nižší priemer, na jednej funkcii bol tento rozdiel podstatne väčší.

Medián s nižšou hodnotou dosahoval nový algoritmus podobne ako u funkcií, čo sa týkalo priemernej hodnoty. Nižší medián však dosiahol už iba 4-krát. Ako v predchádzajúcich prípadoch boli dosiahnuté rovnaké hodnoty mediánu na oboch algoritmoch. V ďalších prípadoch boli výsledky porovnateľné, až na jeden.

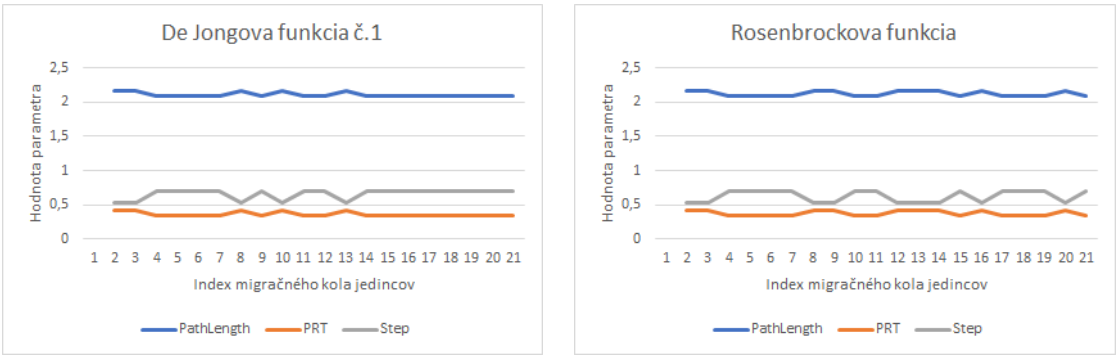
Smerodajná odchýlka vyšla v polovici prípadov s menšími hodnotami a v polovici prípadov s väčšími hodnotami.

7.4 Porovnanie výsledkov nastavenia jednotlivých parametrov

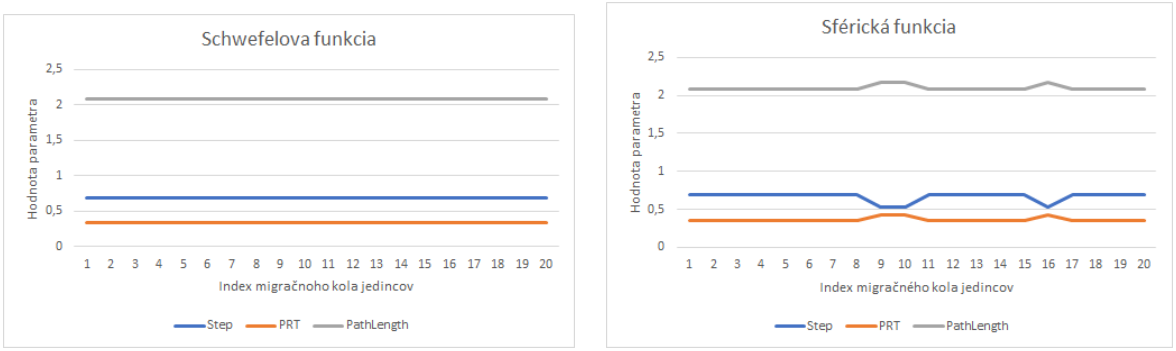
Neurónová sieť nastavovala jednotlivé parametre veľmi podobne. Hodnoty, ktoré boli nastavené sa líšia v jednom behu iba minimálne. Na niektorých funkciách, ako napr. Griewankova (Obr. 13) či Schwefelova (Obr. 16) boli nastavené parametre počas celého behu fixné. Na Ackleyho funkcii (Obr. 14) môžeme vidieť, že jeden parameter počas behu menila, zvyšné dva boli konštantné. Na zvyšných funkciách (Rastrirginova (Obr. 14), De Jongova (Obr. 15), Rosenbrockova(Obr. 15), Sférická (Obr. 16), Styblinski-Tang (Obr. 17), Sum Squares (17) a Zakharova (Obr. 13)) sa všetky parametre menili, hoci rozdiely medzi nimi boli iba minimálne, krivka zmenených hodnôt ukazuje, že vo väčšine prípadov všetky parametre menila spoločne. Znázornených je prvých 20 migračných kôl, vždy migračné kolo aktuálneho jedinca, tento priebeh bol však počas celého behu veľmi podobný. Ukázalo sa, že aj pri fixnom nastavení parametrov ich dokázala sieť zvoliť lepšie než u pôvodného nastavenia pre SOMA algoritmus, kde na Griewankovej funkcii dosiahla celkové minimum. Hodnoty, ktoré sa počas behu menili a boli prispôsobované sa ukázali byť účinné napr. Sférickej či Zakharovej funkcii, kde dosiahol algoritmus SOMANN všetky porovnávané hodnoty lepšie.



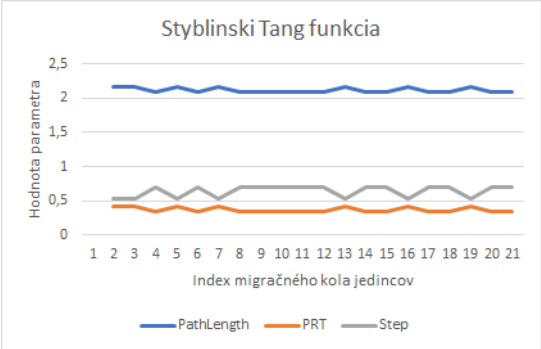
Obr. 14: Nastavovanie parametrov neurónovou sieťou na Ackleyho a Rastirginovej funkcii

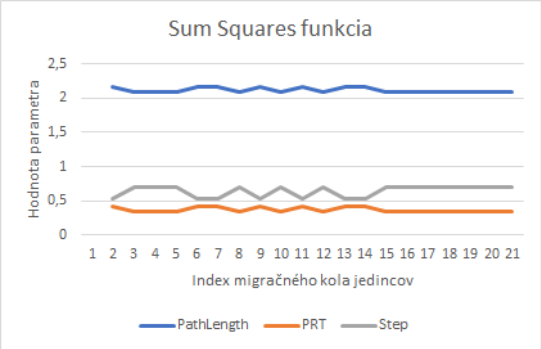


Obr. 15: Nastavovanie parametrov neurónovou sieťou na prvej De Jongovej a Rosebrockovej funkcii

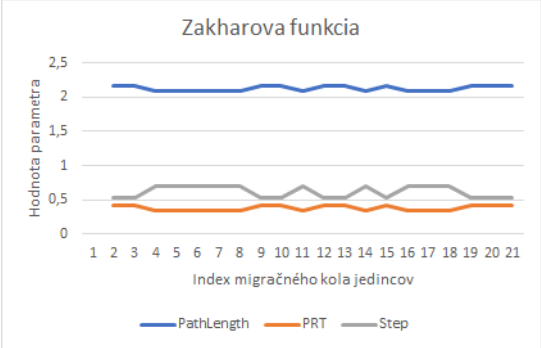


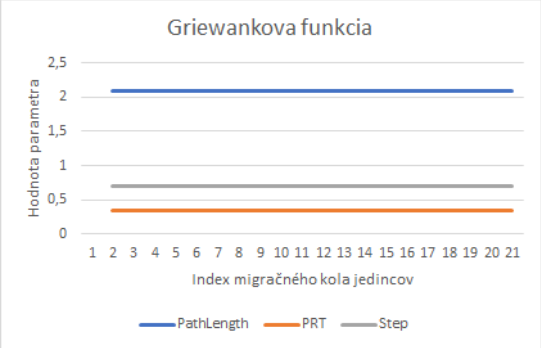
Obr. 16: Nastavovanie parametrov neurónovou sieťou na Schwefelovej a Sférickej funkcii





Obr. 17: Nastavovanie parametrov neurónovou sieťou na Styblinski-Tang a Sum Squares funkcii





Obr. 18: Nastavovanie parametrov neurónovou sieťou na Zakharovej a Griewankovej funkcii

8 Záver

Na záver by som sa chcela zhodnotiť výsledky porovnania pôvodného SOMA algoritmu a SOMANN algoritmu. SOMANN algoritmus oproti pôvodnému algoritmu dosahoval buď podobné alebo lepšie výsledky, jedine na Rastirginovej funkcii si nedokázal poradiť, na ostatných funkciách dosahoval podobné hodnoty. Na funkciách ako prvá De Jongova funkcia, Sum Squares či Zakharova funkcia SOMANN algoritmus dosiahol všetky porovnávané hodnoty nižšie. Dá sa povedať, že závisí od toho, ako je sieť pre danú funkciu naučená. Hodnoty, ktoré sieť nastavovala počas behu algoritmu boli veľmi podobné, niekedy sa vôbec nelíšili a hodnota bola konštantná. Pokiaľ boli tieto hodnoty rovnaké, sieť dokázala v niektorých prípadoch nájsť vhodnejšie fixné parametre, než ktoré boli použité v SOMA algoritme.

Tieto hodnoty však veľmi silno záviseli na tom, aká sieť bola zvolená, ako dlho sa učila, algoritmus je taktiež silno závislý na tom, ako dopadne najlepší jedinec, ktorý je sieti nastavený. Závisí aj od požadovanej funkcie a toho, ako je sieť natrénovaná na danú funkciu. Hodnoty boli ovplyvnené aj tým, že som používala jednu naučenú sieť na všetky testovacie funkcie, sieť teda nebola individuálne prispôsobená na každú funkciu zvlášť.

Dalo by sa povedať, že vplyv na výsledky u jednotlivých testovacích funkcií má unimodálnosť a multiomodálnosť týchto funkcií. Mohli sme si všimnúť, že najhorší výsledok SOMANN algoritmus dosiahol na funkciách Rastirginova či Schwefelova. Naopak na Sférickej, prvej De Jongovej či Zakharovej funkcii dosiahol algoritmus SOMANN oproti pôvodnému algoritmu všetky výsledné hodnoty nižšie.

Naskytuje sa teda priestor pre vylepšenie či zdokonalenie pre sieť, ktorá bude adaptovaná na konkrétnu testovaciu funkciu či zmenou optimalizačného algoritmu napr. za algoritmus MOSOMA, ktorý je určený na optimalizáciu viacerých účelových funkcií naraz.

Literatúra

1. ZELINKA, Ivan. *New optimization techniques in engineering*. SOMA—self-organizing migrating algorithm. Springer, 2004.
2. VONDRÁK, Ivo. Neuronové sıtě. *Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky*. 2009.
3. BABU, G Sateesh; SURESH, Sundaram. Meta-cognitive neural network for classification problems in a sequential learning framework. *Neurocomputing*. 2012, roč. 81, s. 86–96.
4. ALARCON-AQUINO, Vicente; BARRIA, Javier A. Multiresolution FIR neural-network-based learning algorithm applied to network traffic prediction. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*. 2006, roč. 36, č. 2, s. 208–220.
5. KRASILENKO, Vladimir; LAZAREV, Alexander; GRABOVLYAK, Sveta. Design and simulation of a multiport neural network heteroassociative memory for optical pattern recognitions. In: *Optical Pattern Recognition XXIII*. 2012, zv. 8398, 83980N.
6. DARWISH, Ashraf. Bio-inspired computing: Algorithms review, deep analysis, and the scope of applications. *Future Computing and Informatics Journal*. 2018, roč. 3, č. 2, s. 231–246.
7. BACK, Thomas. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.
8. PRICE, Kenneth V. Differential evolution. In: *Handbook of Optimization*. Springer, 2013, s. 187–214.
9. EBERHART; YUHUI SHI. Particle swarm optimization: developments, applications and resources. In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 2001, zv. 1, 81–86 vol. 1.
10. YANG, Xin-She. *Nature-inspired Metaheuristic Algorithms*. Luniver Press, 2010. ISBN 978-1-905986-28-6.
11. KARABOGA, Dervis; AKAY, Bahriye. A comparative study of artificial bee colony algorithm. *Applied mathematics and computation*. 2009, roč. 214, č. 1, s. 108–132.
12. SCHUTTE, Jaco F; REINBOLT, Jeffrey A; FREGLY, Benjamin J; HAFTKA, Raphael T; GEORGE, Alan D. Parallel global optimization with the particle swarm algorithm. *International journal for numerical methods in engineering*. 2004, roč. 61, č. 13, s. 2296–2315.
13. MIRJALILI, Seyedali; MIRJALILI, Seyed Mohammad; LEWIS, Andrew. Grey wolf optimizer. *Advances in engineering software*. 2014, roč. 69, s. 46–61.

14. YAZDANI, Maziar; JOLAI, Fariborz. Lion optimization algorithm (LOA): a nature-inspired metaheuristic algorithm. *Journal of computational design and engineering*. 2016, roč. 3, č. 1, s. 24–36.

15. ZELINKA, Ivan; OPLATKOVÁ, Zuzana; OŠMERA, Pavel; ŠEDA, Miloš; VČELAŘ, František. *Evolučnı vpočetnı techniky: principy a aplikace*. BEN, 2008.

16. DOS SANTOS COELHO, Leandro; MARIANI, Viviana Cocco. *An efficient cultural self-organizing migrating strategy for economic dispatch optimization with valve-point effect.* Energy Conversion and Management. 2010.

17. DEEP, K.; DIPTI. A new hybrid Self Organizing Migrating Genetic Algorithm for function optimization. In: *2007 IEEE Congress on Evolutionary Computation*. 2007, s. 2796–2803.

18. DAVENDRA, Donald; ZELINKA, Ivan; BIALIC-DAVENDRA, Magdalena; SENKERIK, Roman; JASEK, Roman. Discrete self-organising migrating algorithm for flow-shop scheduling with no-wait makespan. *Mathematical and Computer Modelling*. 2013, roč. 57, č. 1-2, s. 100–110.

19. KADLEC, Petr; RAIDA, Zbyněk. Multi-objective Self-organizing Migrating Algorithm. In: *Self-Organizing Migrating Algorithm: Methodology and Implementation*. Ed. DAVENDRA, Donald; ZELINKA, Ivan. Springer International Publishing, 2016. ISBN 978-3-319-28161-2.

20. SINGH, Dipti; AGRAWAL, Seema. Self organizing migrating algorithm with quadratic interpolation for solving large scale global optimization problems. *Applied Soft Computing*. 2016, roč. 38, s. 1040–1048.

21. SKANDEROVA, Lenka; FABIAN, Tomas; ZELINKA, Ivan. Self-adapting self-organizing migrating algorithm. *Swarm and Evolutionary Computation*. 2019, roč. 51, s. 100593.

22. TOMASZEK, Lukas; ZELINKA, Ivan; CHADLI, Mohammed. On the Leader Selection in the Self-Organizing Migrating Algorithm. In: *MENDEL*. 2019, zv. 25, s. 171–178. Č. 1.

23. DAYHOFF, Omid Omidvar Judith. *Neural Networks and Pattern Recognition*. Academic Press, 1997. ISBN 9780125264204.

24. WAIBEL, Alex. *Modular Construction of Time-Delay Neural Networks for Speech Recognition*. 1989.

25. SU, Bolan; LU, Shijian. *Computer Vision – ACCV 2014*. Accurate Scene Text Recognition Based on Recurrent Neural Network. Ed. CREMERS, Daniel; REID, Ian; SAITO, Hideo; YANG, Ming-Hsuan. Springer International Publishing, 2015.

26. COCHOCKI, A.; UNBEHAUEN, Rolf. *Neural Networks for Optimization and Signal Processing*. 1st. USA: John Wiley Sons, Inc., 1993. ISBN 0471930105.

27. NAMPHOL, A.; CHIN, S. H.; AROZULLAH, M. Image compression with a hierarchical neural network. *IEEE Transactions on Aerospace and Electronic Systems*. 1996-01, roč. 32, č. 1, s. 326–338. ISSN 2371-9877. Dostupné z DOI: 10.1109/7.481272.

28. K., Gurney. *An introduction to neural networks*. UCL Press, 1997. ISBN 1-85728-673-1 HB.

29. ZELINKA, Ivan. Metody umělé inteligence. *Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky*. 2019.

30. CROSS, Simon S; HARRISON, Robert F; KENNEDY, R Lee. Introduction to neural networks. *The Lancet*. 1995, roč. 346, č. 8982, s. 1075–1079.

31. PANCHAL, Gaurang; GANATRA, Amit; KOSTA, YP; PANCHAL, Devyani. Behaviour analysis of multilayer perceptrons with multiple hidden neurons and hidden layers. *International Journal of Computer Theory and Engineering*. 2011, roč. 3, č. 2, s. 332–337.

32. ZILLY, Julian Georg; SRIVASTAVA, Rupesh Kumar; KOUTNIK, Jan; SCHMIDHUBER, Jürgen. Recurrent Highway Networks. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. Sydney, NSW, Australia: JMLR.org, 2017, s. 4189–4198. ICML'17.

33. QIN, Yao; SONG, Dongjin; CHEN, Haifeng; CHENG, Wei; JIANG, Guofei; COTTRELL, Garrison. A dual-stage attention-based recurrent neural network for time series prediction. *arXiv preprint arXiv:1704.02971*. 2017.

34. CHUNG, Junyoung; GULCEHRE, Caglar; CHO, KyungHyun; BENGIO, Yoshua. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. 2014.

35. LAWRENCE, Steve; GILES, C Lee; TSOI, Ah Chung; BACK, Andrew D. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*. 1997, roč. 8, č. 1, s. 98–113.

36. HOSSEINI-ASL, Ehsan; KEYNTON, Robert; EL-BAZ, Ayman. Alzheimer's disease diagnostics by adaptation of 3D convolutional network. In: *2016 IEEE International Conference on Image Processing (ICIP)*. 2016, s. 126–130.

37. KARIM, Fazle; MAJUMDAR, Somshubra; DARABI, Houshang; CHEN, Shun. LSTM fully convolutional networks for time series classification. *IEEE access*. 2017, roč. 6, s. 1662–1669.

38. PEROL, Thibaut; GHARBI, Michaël; DENOLLE, Marine. Convolutional neural network for earthquake detection and location. *Science Advances*. 2018, roč. 4, č. 2, s. e1700578.

39. OLORUNDA, O.; ENGELBRECHT, A. P. Measuring exploration/exploitation in particle swarms using swarm diversity. In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. 2008, s. 1128–1134.

40. SURJANOVIC, S.; BINGHAM, D. *Virtual Library of Simulation Experiments: Test Functions and Datasets* [Retrieved May 6, 2020, from `http://www.sfu.ca/~ssurjano`].

41. LUO, Fengji; ZHAO, Junhua; DONG, Zhao Yang. A new metaheuristic algorithm for real-parameter optimization: Natural aggregation algorithm. *2016 IEEE Congress on Evolutionary Computation (CEC)*. 2016, s. 94–103.